特許貧乏物語

（C）宇山　靖政

１．盗聴はさせない……3
　１．１　盗聴解禁（２０００年４月）……3
　１．２　暑い夏の日（２００１年８月）……4
　１．３　知らないことが大事……5
２．日本の特許庁への特許申請……5
　２．１　意外と大変事務手続き……5
　２．２　特許庁への送信完了（２０００年１２月２１日）……7
　２．３　申請書の書き方……8
　２．４　どこに行ったの９２３００円……9
　２．５　OPEN DESIGN（２００１年６月）……10
　２．６　弁理士さんありがとう……20
　２．７　誘惑の手紙（日本の特許紹介業）……20
３．ヨーロッパ出願（２００１年１２月１８日申請受理）……20
　３．１　ヨーロッパに出した請求項：（２００２年０９月２５日公開）……20
　３．２　HONG KONG PATENT AGENCY（２００３年０１月２４日）……22
４．アメリカ出願……22
　４．１　アメリカ出願完了（２００１年１２月１９日）……22
　４．２　逆転する時間（公開中止）（２００２年５月１日）……23
　４．３　勝利の予感（米国公開）（２００２年９月１２日）……25
５．企業との対応……25
　５．１　日本の会社（２００３年）……25
　５．２　ノキア（２００３年）……25
　５．３　ジェトロに相談（２００４年）……27
６．日本特許庁との論争……29
　６．１　日本特許庁の応答……29
　６．２　拒絶査定（２００５年６月１６日）……29
７．ヨーロッパ特許庁……32
　７．１　論争開始（２００４年８月１８日）……32
　７．２　弁理士の反論案（２００４年１２月６日）……71
　７．３　勝利！（２００６年６月１５日）……74
　７．４　審査官のアドバイス……74
　７．５　ヨーロッパでの経過……74
　７．６　取得の表示（２００７年１月１１日）……75
　７．７　バレンタインデー（２００７年２月１４日）……75
　７．８　イギリス特許庁（２００７年３月１日）……75
　７．９　特許証書到着（２００７年３月８日）……75
　７．１０　フィンランド特許庁（２００７年５月２４日）……75
　７．１１　ドイツ特許庁（２００７年５月３１日）……75
８．アメリカ決戦……76
　８．１　最終反論（原案）（２００６年７月３１日）……76
　８．２　反論提出（修正済み）（２００６年１０月７日）……128
　８．３　決戦延期（２００６年１１月７日）……164
　８．４　勝利の旗は閃かん（２００７年１月１７日）……165
　８．５　許可通知（２００７年３月１日）……165
　８．６　特許番号決定……165
９．暗号の輸出規制……165
　９．１　法律の規制……165
　９．２　経済産業省へのお願い（２００６年８月２９日）……168
　９．３　経済産業省の返事（２００６年１２月１４日）……170
　９．４　特許実施許諾（２００７年６月１１日）……170
　９．５　お金を下さい……175
１０．暗号ソフトのソースコード……176
　１０．１　Camellia 暗号ソースコード……176

１０．１．１　暗号化部分 176
１０．１．２　復号化部分 186
１０．１．３　鍵の形式 196
１０．２　AES 暗号ソースコード 196
１０．２．１　暗号化部分 196
１０．２．２　復号化部分 205
１０．２．３　鍵の形式 215
１０．３　Twofish 暗号ソースコード 215
１０．３．１　暗号化部分 215
１０．３．２　復号化部分 259
１０．３．３　鍵の形式 303
１０．４　Serpent 暗号ソースコード 304
１０．４．１　暗号化部分 304
１０．４．２　復号化部分 327
１０．４．３　鍵の形式 352
１０．５　MARS 暗号ソースコード 352
１０．５．１　暗号化部分 356
１０．５．２　復号化部分 377
１０．５．３　鍵の形式 399
１１．　暗号ソフトを作成される方に 399
１１．１　使用できる暗号の種類と特徴： 399
１１．２　暗号ソフトの呼び出しコード 401
１２．価格 401
１３．利用しているフリーソフトについて 401
１４．Protected Communication System (MailJCM)の作成 402
１４．１　開発方針 402
１４．２　参考文献 404
１４．３　教えてグー 404
１４．４　バグ取り 404
１５．IP 電話 404
１５．１　開発方針 404
１５．２　参考文献 404

特許貧乏物語

宇山靖政

１．盗聴はさせない

１．１　盗聴解禁（２０００年４月）

あれは、２０００年４月でした。“日本でも盗聴法ができて、誰でも自由に盗聴して良いようになる。”　新聞をあまり読まない私は、テレビのニュースを聞きながら、盗聴法の内容について、そう信じていました。でも、自分のメールや電話が盗聴されたらいやだなと思っていました。

たしか憲法では、

憲法第二十一条【集会・結社・表現の自由、検閲の禁止、通信の秘密】
１ 集会、結社及び言論、出版その他一切の表現の自由は、これを保障する。
２ 検閲は、これをしてはならない。通信の秘密は、これを侵してはならない。

となっていますが、これからは、自己責任の時代になるのだから、憲法改正で、

憲法第２１条は“通信の秘密は、これを自衛しなくてはならない。”

の様に変わるのだろうなと一人で納得していました。
いまとなれば馬鹿な話です。しかも、できたものは盗聴法ではなく通信傍受法（2000 年 8 月 15 日施行）でした。

通信傍受法では
第 十三条　検察官又は司法警察員は、傍受の実施をしている間に行われた通信であって、傍受令状に記載された傍受すべき通信（以下単に「傍受すべき通信」という。）に該当するかどうか明らかでないものについては、傍受すべき通信に該当するかどうかを判断するため、これに必要な最小限度の範囲に限り、当該通信の傍受をすることができる。
２　外国語による通信又は暗号その他その内容を即時に復元することができない方法を用いた通信であって、傍受の時にその内容を知ることが困難なため、傍受すべき通信に該当するかどうかを判断することができないものについては、その全部の傍受をすることができる。この場合においては、速やかに、傍受すべき通信に該当するかどうかの判断を行わなければならない。

暗号化されていないように見えるが、本当は暗号化されている文章の例は、インターネットで“平文を装った暗号.”で検索すればすぐ見つかります。
これをヒントに私も作ってみました。乱数が並んだビットマップファイルです。もちろん普通の文章に戻せますが、単に乱数を並べただけのファイルと乱雑さにおいて統計的な差がないものになっています。
これが、乱数を並べただけの画像ファイルなのか、暗号文を画像ファイルに変換したものなのかはどうしたら区別できるのでしょうか？私には分かりません。
さらに、駄洒落作成ソフトを組み合わせれば、乱数のビットマップファイルを駄洒落の文章に変換できます。駄洒落を並べたテキストファイルは暗号文なのでしょうか？
作り方は簡単です。８ビットで２５６ですから、２５６個の駄洒落を用意しておいて、乱数列を８ビットごとに区切り、その８ビットを数値として扱いその数字に対応する番号の駄洒落を選んでテキストファイルに書き込む。これによって２５６種類の駄洒落が不規則に並んだ日本語のテキストファイルが得られます。このテキストファイルは暗号文なのでしょうかそれとも本当の乱数列によって決まる駄洒落の列なのでしょうか？

暗号化されているか暗号化されていないか判断できないような形のデータを使った暗号通信は他にも可能です。すると、一見して暗号化されていないように見えるものも、本当は暗号化されている可能性があります。ということは、全ての通信を傍受する必要がある。という結論になります。
それと、方言も外国語のように即時に復元できないものにはいるので、方言で電話すると録音されるのかな。さらに、年寄りにとって若者言葉は完全な暗号です。よって、若い人の電話は全て録音し解析しなくてはなりません。
電話交換機の操作をする人は忙しそうですね。

私は、善良な国民ですから要請があれば暗号解読に協力します。条件は３つ
１．時給５０万円で１日４時間労働、しかもお金は前払い。
２．解読できなくてもお金は返さなくてよい。
３．守秘義務無し、おしゃべり自由。
何しろ、私はおしゃべりですから。

暗号文の盗聴では暗号通信の開発者が解読に協力する事も考えられます。少なくとも私は協力します。従って開発者が提供した暗号以外の暗号アルゴリズムを間に挟んだ多重暗号化が必要になります。開発者の知らない方式も含めて多重に暗号化されていれば、開発者が解読できないとしても無理はない。
開発者は安心して盗聴に協力出来ます。なぜなら解読できなくて当たり前だからです。解読する方法は一つだけあります。送信者と受信者のコンピュータと記録媒体を盗み出してしまうことです。暗号媒体が USB のメモリースティックの場合は相手のポケットや鞄から盗み出せばよい。でも、これだと盗聴する意味が無くなってしまいますね。

自己責任で自分の通信内容の秘密を守る。自分の通信だけは盗聴させない。これを目標に新しい暗号通信の仕組みを考え始めました。
暗号通信の理想型はどんなものなのかについて考えました。
（１）暗号通信網の一部が崩壊しても全体としては崩壊しない。
（２）解読技術の進歩に対抗できる。
（３）開発者にも解けない暗号である。

（１）に関しては、ドイツのエニグマの例が参考になる。U ボートの映画ではドイツの潜水艦から暗号鍵が盗まれて、それが原因でドイツ軍全体の通信が解読されてしまいました。通信相手ごとに暗号化アルゴリズムを変更しなくてはなりません。この問題を解決するヒントは数学の順序対と集合の違いでした。
順序対は中学で勉強した平面座標です。座標（２，３）を持つ点と座標（３，２）を持つ点は別の場所にある。従って（２，３）と（３，２）を別のものと考えなくてはならない。集合は集まりです。２と３のあつまりを２と３からなる集合とよび、｛２，３｝と書きます。｛３，２｝は３と２の集まりですから、内容は同じです。よって、順序対（２，３）と（３，２）は別のものですが、集合｛２，３｝と｛３，２｝は同じものです。B さんと C さんの二人の通信は、一つの集合｛B,C｝に対応させるのではなく、２つの順序対（B→C）と（C→B）の組に対応させるべきです。

（２）に関しては、最新の暗号をすぐに使える構造を与えることと、多重暗号化で対応します。

（３）に関しては、各ユーザーが自由に作成した暗号ソフトを多重暗号化の途中に組み込めるようにしておけばよいのです。

というような目標を現実の通信システムとして実現したものです。
これを実現した電子メールのプログラムを作るのに３年ほどかかりましたが、特許取得にはもっと長い時間（７年と３ヶ月）がかかりました。おかげで、IP 電話や楕円曲線暗号なども作れそうです。引き延ばしてくれたアメリカ特許庁さん、有り難う。

１．２　暑い夏の日（２００１年８月）

２００１年８月、地下鉄丸の内線の新宿御苑前で下車して、地上に出ると暑さで溶け出しそうな新宿御苑通りのアスファルトの上を沢山の車が走っていました。８月の東京ですから当たり前の風景ですが、緑の多い田舎から出てきた私には息苦しくなる風景でした。
暑さのせいかそれとも単なる方向音痴なのか、ファックスで送られてきた地図を見ながら反対方向に歩き始めてしまいました。どこまで行っても目印の銀行が無いのです。
コンビニに入ってクーラーの風を浴びながら少し休憩しました。もう一度地図を見たら反対方向に歩いてしまったことが分かりました。気を取り直して反対方向に歩いていくと目印の銀行があり、その裏の方へ入っていくと弁理士さんの事務所が入っているビルが見つかりました。
麦茶をいただいてから、弁理士さんと話をしました。対応してくれたのは、所長さんと私の特許申請を直接担当してくれる弁理士さんの二人でした。私が持って行ったのは、特許庁に自分で出願した最初のものとＣＱ出版社の月刊誌に掲載したものでした。すぐに言われたことは、“これでは特許はとれません。”の一言でした。

いま（２００７年）考えれば、当然の発言ですが、私は特許の出願についての本を２冊読んでその本の通りに書いたのですから、そんなはずは無いと思いました。考えている通信技術が特許庁の審査官に承認してもらえる様には書かれていないのがその理由だったのでしょう。
どの様な要素を入れて書き直す必要が有るのかを教えてくれました。私は、順序対に対応した暗号通信であることを説明したのですが、すぐには分かってもらえなかったようです。その場で、書き直すための素材の確認をしました。弁理士さんはいろいろアドバイスをくれました。
そのときは、暗号通信ソフトの輸出規制があることは知りませんでしたので、たとえ特許がとれなくてもこの方式を２１世紀の標準的な通信方式にするのだ、との思いを伝えて帰ってきました。

１．３　知らないことが大事

“特許なんて、簡単にとれる。”全く根拠もなくそう信じて始めました。こんなに時間とお金がかかるなんて思ってもいませんでした。もし、知っていたら出願する前に諦めていたでしょう。特許申請したときは、アメリカ、ヨーロッパ、日本の順に結果が出ると思っていました。現実は逆でした。日本、ヨーロッパ、アメリカの順でした。さすが日本の特許庁、いまや世界一スピーディーな特許庁になったようです。
これから、その経過をお話ししたいと思います。馬鹿なやつと笑ってください。さて、その経過を書いたこの本ですが、名前をどうするか悩んでいました。“特許がとれた”と“特許貧乏”どちらが本の題名として適当か周りの人に相談しました。みんなは“特許貧乏”にしなさいと言いました。質問した私を見るなり、全く迷う事無く答えてくれました。自分の現状をみると、まさに特許貧乏です。
２１世紀の秘密通信の方式は、私の順序対方式暗号通信しかない。（英語名は、”Protected Communication System”です。）何の根拠もなくそう信じながら、あり地獄のような特許の世界を突き進みます。
今までの費用は約８００万円（２００７年４月現在です)。アメリカの結論が引き延ばしの作戦で費用が増加しました。アメリカの最後の兵糧攻めで 100 万円ほどよけいにかかってしまいました。
特許の費用だけではありません。暗号の輸出にはもっと大変な法律的な制限があったのです。東京にある経済産業省へ行って相談したり、書類を送れといわれて送ってから３ヶ月が経ちました。いまだに返事はありません。１２月頃には返事を下さいとお願いしたのに無視されているのでしょうか？？
俺の税金で給料をもらえているのが分からないのか！！　日本の税金のほとんどは俺が納めているんだぞ！！
なんて言ってみたいですね。
この法律の制限では、よく注意しないといけません。なにしろ、ヤマハのヘリコプターの事件があったのですから。あいまいな法律は、力を持っている人の見方ですから。
最近、よく言われます。“それだけお金をかけたら世界を何周も出来きるよ！”と。私は英語が嫌いなので、外国に行く事はありえません。私の代わりに”Protected Communication System”に世界中を見てきてもらいたいです。

２．日本の特許庁への特許申請

２．１　意外と大変事務手続き

特許出願の方法を書いた本を２冊読んで、出願文書を書き上げました。そのときは完璧！と信じていました。
自分でコンピュータを使って、特許庁に提出することにしました。提出までに特許庁とやりとりした事務的な書類の名前を列挙します。

１．識別番号付与請求書（平成１２年１１月１７日）（自分で作成）
　これは、特許庁がお客様を管理するために、お客様番号を利用しているので、お客様登録をして特許庁からお客様番号をもらう手続きです。この書類を特許庁に送ります。
２．識別番号通知（平成１２年１１月２４日作成）
　特許庁から郵便で、貴方のお客様番号はこれです、とのお知らせが来ます。
３．電子出願プログラム CD-ROM 交付請求書（平成１２年１２月３日）（自分で作成）
　通信ソフトの入った CD を下さいとのお願いを特許庁に提出しました。
４．予納届（平成１２年１２月６日）（自分で作成）
　識別番号をもとにして、特許手続きに関する料金を前もって納入出来るようにするのです。

５．電子情報処理組織を使用した特定通知等の受領希望届（平成１２年１２月６日）（自分で作成）
　連絡はコンピュータを使った通信で下さい。とのお願いです。
６．電子情報処理組織使用届（平成１２年１２月６日）（自分で作成）
　特許庁のデータベースに自分のコンピュータを接続させてください。とのお願いです。
７．予納台帳番号通知（平成１２年１２月１１日作成）
　特許庁から貴方の予納台帳番号（銀行の口座番号のようなもの）のお知らせが来ました。この番号の口座にお金を振り込みます。ただし、特許証書を郵便局で買ってそれを張り付けた書類を特許庁に送ります。
８．電子情報処理組織接続確認開始通知（平成１２年１２月１２日作成）
コンピュータを特許庁のデータベースに接続できる事になりました。
９．電子出願プログラム CD-ROM 受領書（平成１２年１２月１７日）（自分で作成）
　特許庁のデータベースに接続するには、特別な通信ソフトが必要です。その通信ソフトが入っている CD~ROM が届きました。
１０．予納書提出（平成１２年１２月１８日）（自分で作成）
　特許申請にはお金がかかります。料金をあらかじめ納入します。金額は特許請求項の数で決まります。私の場合は２１０００円でした。２１０００円の特許印紙を貼って特許庁に郵送しました。
１９日に届いて、口座に入金されました。
１１．受領書（特許願）（平成１２年１２月２１日）（自分で作成）
　コンピュータで特許庁に特許申請のデータを送信しました。すぐに受領書がコンピュータ通信によって送られて来ました。一安心です。（実は、同じ日付でアメリカの特許庁によく似た特許を出願していた人がいました。）それを印刷して大切に保管しました。もちろん特許庁に有るデータは時刻と共にきちんと保存されているから心配ないのですが。紙にして安心しました。
１２．予納残高通知書（平成１３年1月４日作成）
　これは、特許印紙２１０００円を平成１２年１２月１９日に受け取って口座に入れました。そのお金は、平成１２年１２月２１日の特許出願費用として使われたので残高は０円になりました。との内容でした。
１３．特定通知等の受領に関する接続通知（平成１３年1月４日作成）
　コンピュータでいろいろな書類が受け取れます。という特許庁からのお知らせです。
１４．予納書提出（平成１３年3月５日）（自分で作成）
　審査請求のために、９２３００円の特許印紙を貼って書類を郵送しました。
このころは、すぐに審査してもらえば、すぐに特許認定されると信じていました。しかし、このお金は消えてしまうことになります。理由は後ほど明らかになりますが、自分で特許申請をされる方は特に注意して下さい。返してはくれません。
１５．予納残高通知書（平成１３年4月２日作成）
　９２３００円を受け取り、口座に入金しましたとのお知らせが来ました。
１６．受領書（平成１３年4月４日）
　出願審査請求を受け取りましたとのお知らせが来ました。
１７．予納残高通知書（平成１３年5月１日作成）０円
　審査請求をしたので、その料金として９２３００円をいただきましたとのお知らせです。
私は、出願から１年以内での再提出をしました。この再提出は新しい特許出願の扱いを受けます。
審査請求したのは古い特許出願です。それは審査してもらえることなく新しい特許出願の審査をしてもらうことになりました。９２３００円は古い出願に関する費用であり、そのお金は返しません。新しい出願では別途審査料金を払って下さいとのことでした。審査請求は良く読み返して訂正しないとなってから払って下さい。（返してください９２３００円。お願いします特許庁さま。）
１８．予納届失効通知（平成１７年5月２日作成）
　口座は使えなくなったらしい。

連絡先は
〒　100-8915
東京都千代田区霞が関３丁目４番３号
特許庁長官　様
電話：０３－３５８１－１１０１
です。

　電話での対応はとても親切で分かりやすい内容でした。特許庁の電話対応の皆さんご苦労さまです。

それに引き替え審査官の石頭！　まるで最初から､否定して認定しないぞ､と思っているかのような対応です。
　実際の手続きでは困ったことがたくさん起きました。まずは特許印紙（何だ？？）どこに売っているのだ？結局近くの郵便局で買えたのですが、郵便局に特許印紙が売っているなんて知りませんでした。収入印紙を大きくしたようなものです。
　当時は、電話回線を ISDN 回線にすることが必要でした。回線の契約を変えなくてはなりません。NTT に電話をして回線の変更をしました。ルーターはキャンペーン中のものを見つけて無料で手に入れました。回線が何とかつながりやっと特許庁に送信できました。
　とにかく、NTT や特許庁に電話をかけまくって何とかやり終えました。たくさんの方にお世話になりました。
　出願後、すぐに同様の内容を雑誌に投稿して発表しようと考えていました。
CQ 出版社との交渉の結果、了解してもらえました。特許の内容が公開される前に雑誌に載せてもらえました。原稿料が入ってちょっと嬉しかったです。しかし、反響はほとんど有りませんでした。
　誉めてくれたのは、法政大学の先生１人だけ。九州大学の先生は読んでくれたはずなのに何もいってはくれませんでした。

２．２　特許庁への送信完了（２０００年１２月２１日）

　２０００年１２月２１日、やっと特許庁に送信しました。識別番号付与請求書（平成１２年１１月１７日）から１ヶ月あまりたって、やっとコンピュータを特許庁に接続できました。すぐにデータを送信しました。最初のページは次のようになっています。

【書類名】　　特許願
【整理番号】　　ＹＵ２０００－０１
【あて先】　　特許庁長官　殿
【国際特許分類】　　Ｈ０４Ｋ　　１／００
【発明者】
　【住所又は居所】　　＊＊＊＊＊＊＊＊＊
　【氏名】　　特許　太郎
【特許出願人】
　【識別番号】　　３０００＊＊＊＊＊
　【氏名又は名称】　特許　太郎
【手数料の表示】
　【予納台帳番号】　１２＊＊＊＊
　【納付金額】　　２１０００
【提出物件の目録】
　【物件名】　　明細書　1
　【物件名】　　図面　　1
　【物件名】　　要約書　1
【書類名】　　明細書
【発明の名称】　　順序対方式暗号通信

　こも文書を作るのに何度も特許庁に電話しました。何しろお客様ですから親切に教えてくれます。お金を取ったら返さないし、特許なんて認めないぞなんて態度は絶対にとりません。あくまでも親切丁寧です。

　電話での質問事項は
１．【0001】を書く場所はどこか。
２．[　] と【　】は同じなのか？
３．文書の拡張子は、htm にするのか、html　にするのか？
４．閲覧の費用６００円は予納するのか、テストのときも６００円必要か？
５．審査の順番はお金を納入した順序か？
６．HTML の文字コードはシフトジスなのかジスなのか？
などなど。

とにかく聞きまくって、この文書が出来ました。

２．３　申請書の書き方

ここが重要です。解説書の通りに書いたのでは、特許はとれません。将棋で言えば定跡が書いてあります。定跡だけでは実践で勝てません。

特許出願の文書は特許請求項と詳細部分からなります。解説書には、詳細部分の内容を含めて判断するから、となっているのですが、いざ、審査にはいると、請求項が抽象的に書いてあると特許の対象ではないとされてしまいます。

私は、請求項を抽象的、一般的に書いてしまいました。（広く特許を取ろうと欲張ったのです。）
詳細部分には、具体的に書いてあったのですが、請求項をみて、思想は特許の対象ではないと言われました。私は、思想のない特許なんて意味がないと自分に言い聞かせました。

最初に書いた請求項：
【発明の名称】　　順序対方式暗号通信
【特許請求の範囲】
【請求項１】 記憶装置と演算装置を利用できる情報機器による暗号を利用した秘密通信の方法のうちで、送信者（Ｓ）と受信者（Ｒ）の順序対（Ｓ,Ｒ）に対して一意的に決まる暗号化方式（Ｃsr）と復号化方式（Ｐsr）に従って暗号を利用した通信を自動的に行うもので、順序対（Ｓ,Ｒ）に対応する暗号化方式を受信者（Ｒ）が前もって自由に指定しておくことができる秘密通信の方法。
【請求項２】 暗号化の指定欄と復号化の指定欄がある識別記号（メールアドレスや電話番号）一覧を備え、送信時には情報の受信者が前もって指定した方式により自動的に暗号化を行い、受信時には送信者に対応する復号化方式により自動的に復号化することで、請求項１の秘密通信の方法を実現する通信用ソフトウエア。
【請求項３】 暗号ソフトの形式のうち、暗号化部分と復号化部分に分割でき、暗号化部分の自由な再配布を認めるが、復号化部分の自由な再配布は認めない形式。
【請求項４】 有料での情報配信の方法で、上記の請求項１，２，３の利用によって少なくとも顧客の要求する以上の強度を持つ暗号化によって情報を配信する方法。

簡潔ですが、暗号通信を送信者と受信者の順序対ごとに行う。事を示しています。
アリスからボブへの通信は、RSA でおこない、ボブからアリスへの返信はカメリアで暗号化する。
そんな事が書いてあるわけです。

さいわいにして、詳細部分にはかなり詳しく書いてあったので、請求項の変更が出来ました。

弁理士さんが修正した請求項：
【発明の名称】　　秘密通信方法
【特許請求の範囲】
【請求項１】 記憶装置と演算装置を利用できる情報機器による暗号を利用した秘密通信の方法のうちで、送信者と受信者の順序対に対して一意的に決まる暗号化方式と復号化方式に従って暗号を利用した通信を自動的に行うもので、順序対に対応する暗号化方式を受信者が前もって自由に指定しておくことができる秘密通信の方法。
【請求項２】 暗号化の指定欄と復号化の指定欄がある識別記号一覧を備え、送信時には情報の受信者が前もって指定した方式により自動的に暗号化を行い、受信時には送信者に対応する復号化方式により自動的に復号化することで、請求項１の秘密通信の方法を実現する秘密通信用ソフトウェア。
【請求項３】 暗号ソフトの形式のうち、暗号化部分と復号化部分に分割でき、暗号化部分の自由な再配布を認めるが、復号化部分の自由な再配布は認めない形式。
【請求項４】 有料での情報配信の方法で、上記の請求項１，２，３の利用によって少なくとも顧客の要求する以上の強度を持つ暗号化によって情報を配信する方法。
【請求項５】 ネットワークを通じた通信機能を有するパーソナルコンピュータ等の情報処理装置に適用される秘密通信用ソフトウェアであって、
送信予定ファイルについての所定の暗号化指示入力があったときには、そのとき指定された送信先データと、送信先データと使用暗号化プログラムとが予め対応づけられた参照テーブルとに基づき、前記情報処理装置の有する記憶手段に予め記憶された複数種の暗号化プログラムから使用すべき暗号化プログラムを自動

特定し、当該暗号化プログラムにしたがって、前記送信予定ファイルを自動暗号化する機能と、
受信済み暗号化ファイルについての所定の復号化指示入力があったときには、当該暗号化ファイルの送信元データと、送信元データと使用復号化プログラムとが予め対応付けられた参照テーブルとに基づき、記憶手段に記憶された複数種の復号化プログラムから使用すべき復号化プログラムを自動特定し、当該復号化プログラムにより前記受信済み暗号化ファイルを自動復号化する機能と、
を具備する秘密通信用ソフトウェア。

【請求項６】　参照テーブルとして情報処理装置に別途記憶されている電子メールソフトのアドレス帳機能を利用するとともに、自動暗号化された送信予定ファイルを当該電子メールソフトによる電子メールに添付してそのとき指定された送信先へと自動送信するための電子メールソフト互換機能を有する、ことを特徴とする請求項５に記載の秘密通信用ソフトウェア。

【請求項７】　参照テーブルにおいては、１つの送信先データに対して複数種の暗号化プログラムが適用順データとともに対応づけ可能とされており、当該対応付けがなされている送信先データで特定される送信予定ファイルは、それら複数種の暗号化プログラムを適用順データに基づき順番に使用することにより多重暗号化され、
また、参照テーブルにおいては、１つの送信元データに対して複数種の復号化プログラムが対応付け可能とされており、当該対応付けがなされている送信元データで特定される復号化ファイルは、それら複数種の復号化プログラムを適用順データに基づき順番に使用することにより復号化が行われる、ことを特徴とする請求項５又は６に記載の秘密通信用ソフトウェア。

【請求項８】　参照テーブルには、各送信先データまたは送信元データ毎に、電子メール本文と電子メール添付ファイルとのそれぞれについて別個に複数の暗号化プログラムまたは復号化プログラムを対応付け可能とされている、ことを特徴とする請求項７に記載の秘密通信用ソフトウェア。

【請求項９】　適用順データは、予め定められた規則に基づき適宜自動変更される、ことを特徴とする請求項７または８に記載の秘密通信用ソフトウェア。

【請求項１０】　ナンバーディスプレイ等の発信者通知機能を有する電話機であって、
暗号化プログラムおよび復号化プログラムとを入力する手段と、
発信先データと任意の暗号化プログラムとを対応づけて暗号化参照データとして予め記憶させるための手段と、発信元データと任意の復号化プログラムとを対応づけて復号化参照データとして予め記憶させるための記憶手段と、そのときの通話に係る発信先データと前記暗号化参照データとに基づきその通話に際して使用する暗号化プログラムを特定し、当該暗号化プログラムを使用して送信すべき音声信号を適宜暗号化して送信する手段と、そのときの通話に係る発信者データと前記復号化参照データとに基づきそのとき使用する復号化プログラムを特定し、当該復号化プログラムを使用して順次受信される音声信号を適宜復号化して音声出力する手段と、を有する、電話機。

【請求項１１】　参照テーブルにおいては、１つの発信先データに対して複数種の暗号化プログラムが対応づけ可能とされており、当該対応付けがなされている発信先への送信される音声信号は、それら複数種の暗号化プログラムに基づき多重暗号化され、
また、参照テーブルにおいては、１つの発信元データに対して複数種の復号化プログラムが対応付け可能とされており、当該対応付けがなされている発信元から送信されてくる音声信号は、それら複数種の復号化プログラムに基づき多重復号化される、ことを特徴とする請求項１０に記載の電話機。

と変更しました。
変更内容は、詳細な説明の部分に書いてあったものを請求項に変えたのです。ただし、詳細な説明に書いてないことは付け加えられないのだそうです。なんと言っても名前がかっこよくなりました。

請求したらすぐ OK と思っていましたが、そんなに甘くはありませんでした。日本の特許庁とのやりとりについては後ほど詳しく書きます。

２．４　どこに行ったの９２３００円

弁理士さんは、なかなか審査請求をしません。公開されるのをじっと待っています。私は自分で出願したときは、すぐに審査請求をしました。１年以内に、変更が出来るので弁理士さんは請求項をより技術的なものにして特許申請をやり直しました。
私は自分で特許申請したとき、すぐに９２３００円を払って審査請求をしました。内容を変更して特許請求をやり直すと、前の特許に関する審査はしないようになるのです。もちろんお金９２３００円は返ってきません。審査もしないで審査料を返さないとは、特許庁の馬鹿。（金返せ！）

２．５　OPEN DESIGN（２００１年６月）

CQ 出版社の OPEN DESIGN (2001 年６月号)に、やっと掲載されました。ＣＱ出版社との交渉は２０００年の１０月頃から始めました。編集の方には大変ご迷惑をかけてしまいましたが、アメリカ特許庁の変な対応を考えると発表しておいて良かったです。

“メール伝送秘密化の鍵!?　21 世紀の秘密通信“

という題名です。

出版社とは、読んでみて良かったら載せてくれる。との約束をとりつけ、特許庁へ申請した後で、特許が公開される前に出版される雑誌に載せてもらう事にしました。ただし、特許庁へ送信する前に内容を編集者に知らせる事はできませんので、内容のないものについての掲載予定を立ててくれました。

内容は

“メール伝送秘密化の鍵!?　21 世紀の秘密通信“　　宇山　靖政

【０００１】
【はじめに】
IT 革命の声は高いのですが、多くの人たちがネットワーク上での情報の安全性に疑問を持っています。今は、安心してネットワークを利用できる状況ではありません。この現状を変えるにはネットワーク上での情報伝達の安全性に対する信頼を確立する必要があります。
これを実現する基礎は２０世紀の暗号技術を代表する言葉、公開鍵暗号でしょう。しかしながら、秘密通信の方式そのものが不完全なために、せっかくの技術が十分に活用されてはいません。２０世紀の暗号技術を十分に活用するための方法が必要だと考えます。この方法がここで提起する順序対方式暗号通信です。
通信の秘密は、暗号化鍵の一部を通信を傍受できる立場のものに与えたままでは実現できません。データを送信する全ての人が自分の欲しいだけの強度を持った暗号化をすることの出来る環境が必要です。この方法を示したものがこれです。
ただし、現在特許申請中ですので安易に作成したに販売したりはしないで下さい。

【０００２】
【２０世紀の秘密通信の現状と問題点】
２０世紀の暗号通信は公開鍵暗号と公開鍵の登録制度に象徴されます。すなわち、暗号技術そのものは発展したのですが、それをどのような形で利用するかに関しては、ローマ時代のシーザーの頃と変わってはいません。それは、特別な情報を暗号化して伝えると考えていることにあります。
インターネットでの通信は、暗号化しないで送信されるものがほとんどです。暗号化されるのは特別な通信の場合だけというのが現状でしょう。これは、ネットワークの上を裸で歩き回る事と同じです。２１世紀は、情報が洋服を着てネットワーク上を移動するようにしたいものです。郵便物を出すときに、葉書にするか、封書にするかは個人が決めます。個人的な情報は封書で送る人が多く、儀礼的、形式的な情報のみが葉書で伝えられます。少なくとも封書で伝える程度の安全策をインターネットでの通信も持つべきでしょう。
暗号化にはコストがかかる。大量の計算を必要とする。これは、現在のコンピュータにとっては昔話にすぎません。いまや、CPU は有り余る力を有しながらそれをグラフィックのみに利用していると言えます。巨大な計算力はすべての通信を暗号化する事ができます。さらに、大容量のハードディスクは規格化された暗号から、個別的な暗号への転換を可能としてくれています。すべての準備は整っているのです。

【０００３】
【順序対方式暗号通信】
知らない人と、秘密通信をすることは商業的な通信以外ではまずあり得ません。暗号化による秘密通信を使ったほうがよいのは、個人的な通信の場合です。気軽に、自由に、最新の暗号技術を使って、知り合い同士が通信を行えるようにするには、暗号の利用に関しての考え方を変えなくてはなりません。情報は、個別的に、個人的に伝えられるべきものですから、送信者と受信者だけで暗号を利用するか否かを決めればよいのです。鍵を登録する必要はありません。こう考えれば、秘密通信の新しい考え方が自然にでてきます。
秘密通信は、個人の普通の通信において自由に個人の責任で行われるべきです。送信者と受信者の間で暗号を利用するか否か、暗号の形式はどのようなものを採用するかなどを直接決定すればよいのです。したがって、情報の発信者（S）と受信者（R）の順序対（S,R）に対して決定される暗号化方式をＣsr、復号化方式をＰsr とすれば、送信時には、送信者と受信者の順序対（S,R）に対して決定される暗号化方式Ｃsr によって送付する情報を自動的に暗号化し、受信時には、送信者と受信者の順序対（S,R）に対して決定される復号化方

式Ｐsr によって自動的に復号化されたものが表示されるようにすればよいのです。
このとき、受信者が暗号方式を決定し、送信者は受信者によって指定された暗号化鍵、暗号化ソフトを使ってその受信者に対する情報を暗号化して送付するのです。受信者の側では、送信者のアドレスと自分のアドレスから自動的に復号化方式を決定し情報を復号化するようにします。
この順序対方式暗号通信で秘密通信を行えば、インターネットの社会もより安全なものになります。この暗号化方式は受信者が自由に決定できる形式ですので、情報の安全性に関する安心感が生まれます。

【０００４】
【順序対方式暗号通信の概略】
送信者と受信者の順序対に対して、暗号化方式と復号化方式を一意的に決定します。この方式に従った暗号通信が行われるように通信用ソフトの機能を拡張します。暗号方式の指定欄を持つアドレス帳を利用して、暗号化や復号化が自動的に行えるようにするのです。
暗号解読技術の発展に合わせて新しい暗号方式に簡単に変更できるような方法を実現するのです。そして情報の受信者が暗号方式を自由に決定できるようにするのです。
暗号ソフトの構造を、暗号化部分と復号化部分の二つにわけ、暗号化部分の自由な再配布を認めるようにします。また、復号化部分の再配布を禁止できるようにしておきます。これによって、暗号ソフトを商業ベースで開発販売する事を経済的に保証できます、優れた暗号ソフトが安定して供給されるようになるのです。
この方式は電話による通信の場合でも利用できます、現在の電話機に記憶装置と演算装置を付け加える必要があります。さらに、いくつかの項目について規格を統一する必要があります。
そこで、現在のハードウエアですぐに実現できるインターネットの電子メールに関する事柄を中心に記述する事にします。

【０００５】
【順序対方式暗号通信の詳細】
以下、順序対方式暗号通信の詳細を記述します。

【０００６】
　【利用される自然法則】
２つのもの　X、Y　に順序を付けて並べる方法は（X、Y）と（Y、X）の２種類であり、これ以外にはありません。この自然法則を利用して、順序対方式暗号通信を実現します。

【０００７】
【実現の形態】
以下のようにすれば、順序対方式暗号通信を実施できます。

【０００８】
【アドレスの一意性とその順序対】
インターネット社会ではメールアドレスやIPアドレスなど個人を特定するための識別記号が存在します。情報の発信者と受信者の２者を特定すれば、二つのメールアドレスから（発信者、受信者）の順序対を決定できるのです。
この順序対に対応して暗号化方式を決定できるような仕組みを作ります。（X、Y）をＸさんからＹさんへの通信とするとき、この時の暗号化方式を受信者であるＹさんが自由に決定できるようにすればよいのです。
さらに、ＸさんからＹさんへ送られる情報が、Ｙさんが事前に決めておいた方式で自動的に暗号化されるようにします。
Ｙさんが信頼する暗号化技術が、Ｘさんから見て信頼できる技術であるとは限りません。ＸさんがＹさんから情報を送ってもらう時は、順序対（Y、X）によって決まる暗号化方式、すなわちＸさんがＹさんに対してしてあらかじめ指定しておいた暗号化方式で暗号化して送信してもらうようにすればよいのです。
これは、次の各項目を考えれば可能となります。

【０００９】
【アドレス帳の拡張】
現在、インターネットでの通信に利用するソフトにはアドレス帳があり、そこには名前、メールアドレス、組織、電話番号などが記述されています。
これを拡張して、暗号化鍵、暗号化ソフト、復号化鍵、復号化ソフトの４項目を追加します。たとえば、つぎのように変更するのです。

秋山氏のアドレス帳
氏名　アドレス　暗号化鍵　暗号化ソフト　復号化鍵　復号化ソフト
Ｘ社　Xcp@kabu　ＫＣax　Ｃax　ＫＰxa　Ｐxa
伊藤　Itou@asn　ＫＣai　Ｃai　ＫＰia　Ｐia
斉藤　Saito@uu　ＫＣas　Ｃas　ＫＰsa　Ｐsa
馬場　Baba@yy　ＫＣab　Ｃab　ＫＰba　Ｐba

【００１０】
【通信時におけるソフトウエアの機能】
ＸさんがＹさんに情報を送るとします。このとき、順序対（Ｘ、Ｙ）が決まりこれに対応する暗号化方式をこの順序対に対して一意的に決定できます。この事実をもとにして、次のように機能しなくてはなりません。

【００１１】
【送信時の機能】
送信する情報が与えられたら、電子メールの宛先を調べて、アドレス一覧の暗号化鍵、暗号化ソフトの項目をチェックします。それが指定されていたら、暗号化鍵と送信内容を引数として、指定された暗号化ソフトを呼び出し送信する内容を暗号化します。
宛先のアドレスと送信元のアドレスのみを記述した空のメールをつくり、暗号化したものを添付ファイルとして送信します。
もちろん、アドレス帳に記載されていない相手や、アドレス帳に記載されていても暗号化項目が指定されていない場合は暗号化しないでそのままメールを送ることにします。
送信時の機能は、図１のようになります。

【００１２】
【受信時の機能】
受信時には、発信者のメールアドレスを自分のアドレス帳から探し、その発信者の項目に復号化鍵と復号化ソフトが与えられていたら、復号化鍵と添付ファイルを引数として復号化ソフトを呼び出し、復号化を行います。そして暗号化されていない情報を表示します。
もちろん、アドレス帳に記載されていない相手や、アドレス帳に記載されていても復号化項目が指定されていなければ復号化しないでそのままメールを開きます。このような機能をもったソフトウエアまたはハードウエアを作成します。
受信時の機能は、図２のようになります。

【００１３】
【使用できる暗号の種類と特徴】
暗号には、大きく分けて秘密鍵方式と公開鍵方式の２種類がありますが、どちらも利用可能です。社会的な評価が高く、専門家が有効と認めるような暗号ソフトの種類は１００種類程度でしょう。
暗号化鍵と暗号化ソフトが統一された形で実現されたものは、実用上問題が生じます。それは、暗号化鍵のサイズは１Ｋバイト程度でしょうが、統一されたものは１Ｍバイトから１０Ｍバイト程度になります。このサイズのものを１００００人分保存するには１００Ｇバイトが必要になり、１０Ｇバイト程度のハードディスクでは、困難となります。
共通の暗号化ソフトが利用できるならば、暗号化鍵が１００００人分あってもそのサイズは１０Ｍバイト程度ですむのです。
理論上は、暗号化鍵と暗号化ソフトを分離しなくても良いのですが、ハードウエアの現状に適しているとの理由で、暗号化鍵と暗号化ソフトを分離する方式について記述します。
暗号用の商用ソフトは、次のような構造を持つものとして制作されるべきです。
（１）暗号化鍵の作成機能をもち、鍵の自由な配布が認められている。
（２）暗号化ソフトの自由な再配布が認められている。
（３）復号化鍵の作成機能をもつ。
（４）復号化ソフトの再配布を禁止できる。
この、１から４を満たすようになっていれば秘密鍵方式でも、公開鍵方式でもどちらも利用することができます。このような構造で暗号ソフトを制作することは困難ではありません。

復号化ソフトの再配布が禁止できれば、暗号ソフトの利用者がそれぞれにソフトを購入することになりますので、商業的にソフトを制作販売することが可能となります。より優れた暗号ソフトの開発が可能となるので

す。
暗号ソフトを自作しても良いのですが、安全な暗号ソフトを作成するのはかなりの時間と努力を必要とします。暗号ソフトは既にかなりの種類が存在し、日本が世界をリードしている現状があります。社会的評価の高いものを選ぶのが現実的な選択です。

【００１４】
【秘密鍵方式を使う場合】
秘密鍵方式を使うときは、秘密鍵と暗号化ソフトを事前に配布しておくことになります。
具体例としてシーザー暗号方式の場合を示します。
秋山氏は、シーザー暗号を使うことにし、伊藤氏からの情報は暗号化鍵（－１）で、斉藤氏からの情報は２で暗号化してもらうことにしたとします。

秋山氏のアドレス帳
氏名 アドレス 暗号化鍵 暗号化ソフト 復号化鍵 復号化ソフト
X社 Xcp@kabu ＫＣax Ｃax ２５ Ｐ
伊藤 Itou@asn ＫＣai Ｃai １ Ｐ
斉藤 Saito@uu ＫＣas Ｃas －２ Ｐ
馬場 Baba@yy ＫＣab Ｃab ２２ Ｐ

秋山氏は、暗号化鍵と暗号化ソフト（Ｃ）を伊藤氏と斉藤氏に送ります。伊藤氏、斉藤氏は自分のアドレス帳の暗号化欄に登録します。
伊藤氏、斉藤氏のアドレス帳は次のようになります。

伊藤氏のアドレス帳
氏名 アドレス 暗号化鍵 暗号化ソフト 復号化鍵 復号化ソフト
X社 Xcp@kabu ＫＣix Ｃix ＫＰxi Ｐxi
秋山 Akiyama@ss （－１） Ｃ ＫＰai Ｐai
斉藤 Saito@uu ＫＣis Ｃis ＫＰsi Ｐsi
馬場 Baba@yy ＫＣib Ｃib ＫＰbi Ｐbi

斉藤氏のアドレス帳
氏名 アドレス 暗号化鍵 暗号化ソフト 復号化鍵 復号化ソフト
X社 Xcp@kabu ＫＣｓx Ｃｓx ＫＰxs Ｐxs
伊藤 Itou@asn ＫＣsi Ｃsi ＫＰis Ｐis
秋山 Akiyama@ss ２ Ｃ ＫＰas Ｐas
馬場 Baba@yy ＫＣｓb Ｃｓb ＫＰbs Ｐbs

伊藤氏から秋山氏への通信（IBM）は、鍵（－１）で暗号化され、(HAL)となります。
伊藤氏からの暗号文（HAL）は、秋山氏の伊藤氏からの通信に対する鍵（１）で復号化され（IBM）に戻ります。
斉藤氏から秋山氏への通信（IBM）は、鍵（２）で暗号化され、（KDO）となります。
暗号文（KDO）は、鍵（－２）で復号化され、（IBM）に戻ります。
つまり、暗号化鍵の数だけ文字を先に進め、復号化鍵の示す数だけ元に戻す形になっているのです。

【００１５】
【公開鍵暗号方式を使う場合】
暗号化した情報を送ってもらいたい相手全員に公開鍵を送っておけばよいのです。また、暗号化ソフトも同じものを配布しておきます。もちろん、公開鍵方式にもその実現形態はいろいろあるので、相手ごとに別の公開鍵方式による暗号化ソフトを利用することも可能です。
具体例として、離散対数を利用した場合を示します。
素数 p,q を選び、n = pq とします。K = lcm(p-1 , q-1) として、gcd(d,K)=1 となる d を選びます。つぎに、ed = 1 (mod K) , ０＜e＜K,となる整数 e を計算します。
秋山氏は、e , n を公開し、暗号化ソフト（Ｃ）を送付します。
ｄは秘密とし、自分のアドレス帳に n と共に登録しておきます。

秋山氏のアドレス帳

氏名　アドレス　暗号化鍵　暗号化ソフト　復号化鍵　復号化ソフト
X社　Xcp@kabu　ＫＣax　Ｃax　d,n　ＰＳ
伊藤　Itou@asn　ＫＣai　Ｃai　d,n　ＰＳ
斉藤　Saito@uu　ＫＣas　Ｃas　d,n　ＰＳ
馬場　Baba@yy　ＫＣab　Ｃab　d,n　ＰＳ

秋山氏は、暗号化鍵と暗号化ソフトを伊藤氏と斉藤氏に送ったとすれば、
伊藤氏、斉藤氏のアドレス帳は次のようになります。

伊藤氏のアドレス帳
氏名　アドレス　暗号化鍵　暗号化ソフト　復号化鍵　復号化ソフト
X社　Xcp@kabu　ＫＣix　Ｃix　ＫＰxi　Ｐxi
秋山　Akiyama@ss　e, n　Ｃ　ＫＰai　Ｐai
斉藤　Saito@uu　ＫＣis　Ｃis　ＫＰsi　Ｐsi
馬場　Baba@yy　ＫＣib　Ｃib　ＫＰbi　Ｐbi

斉藤氏のアドレス帳
氏名　アドレス　暗号化鍵　暗号化ソフト　復号化鍵　復号化ソフト
X社　Xcp@kabu　ＫＣｓx　Ｃｓx　ＫＰxs　Ｐxs
伊藤　Itou@asn　ＫＣｓI　ＣｓI　ＫＰis　Ｐis
秋山　Akiyama@ss　e , n　Ｃ　ＫＰas　Ｐas
馬場　Baba@yy　ＫＣｓb　Ｃｓb　ＫＰbs　Ｐbs

数　x　を送信するとすれば、
伊藤氏と斉藤氏の持っている暗号化ソフトＣは e,n を使って、
x^e≡c (mod n) を計算し、このｃを秋山氏へ送信します。
受け取った秋山氏は c^d≡x (mod n) によってｘを得ます。すなわち、法ｎでｃのｄ乗を計算して、ｘを得るのです。

【００１６】
【暗号化鍵と暗号化ソフトの配布方法】
情報を送ってもらおうとする相手に、あらかじめ暗号化鍵と暗号化ソフトを所持させる必要があります。直接会ってフロッピーディスクなどに記録したものを渡す。それらを郵送する。又は公開鍵暗号方式などによって暗号化したものを、ネットワークを使って送るなどの方法があります。

【００１７】
【暗号化鍵と暗号化ソフトの登録】
暗号化鍵と暗号化ソフトを受け取った人は、自分のアドレス帳に登録します。
たとえば、秋山氏が伊藤氏との通信で暗号を利用することにしたとしましょう。
　順序対（伊藤、秋山）に対応する暗号化方式がきまります。秋山氏が伊藤氏からの通信を受信するときの暗号化方式をＣ(i,a) とします。
秋山氏は、伊藤氏に会って、暗号化ソフト（Ｃia）と暗号化鍵（ＫＣia）を手渡します。伊藤氏は、アドレス帳の秋山氏の項目に、この暗号化ソフトと暗号化鍵を登録し、暗号化ソフトと暗号化鍵をハードディスク内に記録しておくのです。

伊藤氏のアドレス帳は、
氏名　アドレス　暗号化鍵　暗号化ソフト　復号化鍵　復号化ソフト
X社　Xcp@kabu
秋山　Akiyama@ss　ＫＣia　Ｃia
斉藤　Saito@uu
馬場　Baba@yy

のようになります。

【００１８】
【復号化ソフトの登録】

秋山氏は、伊藤氏からの情報を受信するときに利用する暗号方式を決定したので、これによって決まる復号化鍵（ＫＰia）と復号化ソフト（Ｐia）を、アドレス帳の伊藤氏の欄にある復号化鍵、復号化ソフトの項目に登録します。

秋山氏のアドレス帳は
氏名　アドレス 暗号化鍵　暗号化ソフト　復号化鍵　復号化ソフト
Ｘ社　Xcp@kabu
伊藤　Itou@asn　ＫＰia　　Ｐia
斉藤　Saito@uu
馬場　Baba@yy

のようになります。
このアドレス帳により、秋山氏は伊藤氏からの通信を（Cia）によって暗号化した形で送ってもらうことができます。また、届いた情報は、（Ｐia）によって自動的に解読されるのです。
秋山氏は、同じ暗号化方式、同じ暗号化鍵を伊藤氏以外の人との通信で利用することもできるし、暗号化方式は同じでも、暗号化鍵は変えて他の人との通信を行うこともできます。
また、全く別の方式で暗号化した通信を行うこともできます。なぜなら、暗号化方式とその実現形態は送信者と受信者の順序対で決まるのですから。
　伊藤氏が、秋山氏から送られてくる情報の暗号化に別の方式を選択できることも明らかです。

【００１９】
【復号化ソフト再配布禁止の意義】
暗号化ソフトが、同時に復号化ソフトとして機能することがないようにしておけば、暗号ソフトを購入してこの方式を利用する人は、それぞれ、復号化ソフトを必要とするので、個別に暗号ソフトを購入しなくてはならないことになります。
なぜなら、復号化ソフトは再配布が禁止となっているからです。このことは、暗号ソフトの商業的開発を保証し、より安全な通信のための経済的基盤となります。暗号ソフトの開発には長時間の研究が必要であり、経済的保証がなければ優れた暗号ソフトを開発することはできません。

【００２０】
【暗号ソフトの名前の一意性】
異なる暗号ソフトに同じ名前が付くと、ハードディスクの中に保存するときに問題が生じます。したがって、暗号ソフトの開発者が持つ一意的な識別記号（メールアドレスなど）をその暗号ソフトの名前に使用して、名前の衝突が起きないようにする必要があります。

【００２１】
【商業的利用】
証券会社（Ｘ社）は、顧客に対して有料での情報提供を行うとします。ある顧客が自動車会社に関心を持っていたとしましょう。その顧客にとっては自動車会社の新車開発情報などは自分だけに知らせてもらえるなら、高い値段が付いていても購入する価値のある情報です。
しかし、この自分だけが知りたい情報の伝達が、ネットワーク上で盗聴や改竄を受けるおそれがあるので、今ひとつ不安があります。
そこで、Ｘ社はこのシステムを採用するとしましょう。

Ｘ社の顧客アドレス一覧は
氏名　顧客アドレス　暗号化鍵　暗号化ソフト　復号化鍵　復号化ソフト
秋山　Akiyama@ss　　ＫＣxa　　Ｃxa　　ＫＰax　Ｐax
伊藤　Itou@asn　　ＫＣxi　　Ｃxi　　ＫＰix　Ｐix
遠藤　Endo@kk　　ＫＣxe　　Ｃxe　　ＫＰex　Ｐex
山田　Yamada@yama　ＫＣxy　　Ｃxy　　ＫＰyx　Ｐyx

のようになります。
Ｘ社は、顧客の関心を持つ経済分野の情報を有料で配信するとします。顧客が暗号化方式を決定するのが原則ですが、社会的に評価の高い暗号ソフトを１０種類程度は紹介する必要があります。
逆に、顧客からの注文などを受け取るための暗号方式を決定します。これは顧客の購買力によって方式を変えることもできます。

できれば、異なった暗号化鍵を顧客の数だけ作り出せるものが便利です。しかも、復号化鍵はお客ごとに異なるとしても、復号化ソフトは同じものが利用できるような暗号ソフトがよいでしょう。ここにも、復号化鍵と復号化ソフトを分離できる形のメリットが見いだせます。
価値ある経済情報は自分だけに知らせてほしいと思うし、この情報がネットワーク上で盗聴されたり改竄されたりすることを防ぎたいものです。
この方式は、顧客自身の責任で自分のための情報を保護できます。これによって情報にさらなる付加価値を付けることができるのです。これは、経済的に価値の高い情報を伝えるためには不可欠の方式です。

【００２２】
【個人による利用】
この方式を個人が利用する場合のアドレス帳は次のような形になります。個人で利用する場合は、経済的負担を考える必要があります。
公開鍵暗号などの場合は、同じ暗号化鍵を配布すればすむので、復号化鍵、復号化ソフトは１種類で済ませることもできます。また、商業的な通信と友人間の通信に分けて、別の公開鍵方式を採用することもできます。
秘密鍵方式では、相手ごとに異なる秘密鍵を作成し、これに対応した復号化鍵を自分のアドレス帳の復号化鍵の欄に記録しておくようにします。鍵が異なっても同じ暗号化ソフト、同じ復号化ソフトを使用することもできるので、ハードディスクなどの資源を節約することもできます。もちろん、通信相手ごとに暗号化方式を変えても良いのです。
暗号化ソフトが、暗号化鍵と一体化されていると比較的大きなサイズとなり、これをアドレス帳に記載された人数分だけ自分のハードディスクの中に用意することになります。
人数が増えると、２０Ｇバイト程度のハードディスクでは容量が不足します。そこで、サイズの小さな暗号化鍵は、アドレス帳に記載された人数分だけ用意するにしても、サイズの大きな暗号化ソフトは、共通に利用できることが望ましでしょう。
現実的に有効と認められる暗号ソフトは１００程度と思われます。暗号化ソフトが共通で利用できるなら、ハードディスクに１００種類だけ置いておけばすみます。また、暗号化ソフトだけを、雑誌の付録などで配布することも可能ですので、実際にはそれを入手し、暗号化鍵だけをやりとりすることも、現実的な方法でしょう。

秋山氏のアドレス帳

氏名	アドレス	暗号化鍵	暗号化ソフト	復号化鍵	復号化ソフト
X社	Xcp@kabu	ＫＣax	Ｃax	ＫＰxa	Ｐxa
伊藤	Itou@asn	ＫＣai	Ｃai	ＫＰia	Ｐia
斉藤	Saito@uu	ＫＣas	Ｃas	ＫＰsa	Ｐsa
馬場	Baba@yy	ＫＣab	Ｃab	ＫＰba	Ｐba

伊藤氏のアドレス帳

氏名	アドレス	暗号化鍵	暗号化ソフト	復号化鍵	復号化ソフト
X社	Xcp@kabu	ＫＣix	Ｃix	ＫＰxi	Ｐxi
秋山	Akiyama@ss	ＫＣia	Ｃia	ＫＰai	Ｐai
斉藤	Saito@uu	ＫＣis	Ｃis	ＫＰsi	Ｐsi
馬場	Baba@yy	ＫＣib	Ｃib	ＫＰbi	Ｐbi

このシステムでの通信用ソフトは次のように機能します。
電子メールを送るときには、宛先のアドレスと、発信者のアドレスが最初に決まり、発信者と受信者の順序対が決まることになります。この順序対に対に対応した暗号化方式によって自動的に暗号化されます。
　受信者の方では、発信者のアドレスと宛先である自分のアドレスの順序対から復号化方式を決定できるので、暗号化されて送付された情報を自動的に復号化できます。
たとえば、秋山氏は伊藤氏からの情報を受信するときに利用する暗号化方式を決定します。秋山氏は伊藤氏との通信で使う暗号化鍵（ＫＣia)、暗号化ソフト（Ｃia)、復号化鍵（ＫＰia)、復号化ソフト（Ｐia）を用意し、このうち暗号化鍵（ＫＣia)、暗号化ソフト（Ｃia）に関する情報を事前に伊藤氏に送っておきます。伊藤氏はアドレス帳の秋山氏の項目に、暗号化鍵と暗号化ソフトを登録しておきます。
伊藤氏から秋山氏へのメールは秋山氏が決めた暗号化方式（Ｃia）と暗号化鍵（ＫＣia）によって暗号化される。これを受け取った秋山氏の通信用ソフトは、送信者のアドレスと、自分のアドレスを判別できるので、これによって復号化方式（Ｐia）と復号化鍵（ＫＰia）を自動的に選びこの方式で復号化するのです。
逆に、伊藤氏は秋山氏からの情報を受け取るための暗号化方式を決定します。もちろん、この方式は伊藤氏が自由に決定できます。秋山氏からの情報を受け取るために使う暗号化鍵（ＫＣai)、暗号化ソフト（Ｃai)、復

号化鍵（ＫＰai)、復号化ソフト（Ｐai）を用意します。このうち、暗号化鍵（ＫＣai)、暗号化ソフト（Ｃai）を事前に秋山氏に伝えておきます。
秋山氏から伊藤氏へのメールは伊藤氏が決めた暗号化方式（Ｃai）と暗号化鍵（ＫＣai）によって暗号化されます。受け取った伊藤氏は、秋山氏用の復号化鍵（ＫＰai）と復号化ソフト（Ｐai）を使って解読するのです。

【００２３】
【暗号方式の更新】
暗号技術の進歩は驚くほど早く、暗号解読技術も同時に進歩しています。一つの暗号化方式が安全であるのはせいぜい５年間くらいです。したがって、情報の受信者自身が利用する暗号技術を自由に更新できることが必要です。
この方式では、この変更が情報の受信者の意志で自由にできるのです。その時点での最も優れた暗号技術を選択することも、利用者の経済的状況が許せば可能となります。
新しい暗号方式を利用すると決めた人は、新しい暗号ソフトを購入し、その暗号化部分を自分のアドレス帳に記載されている人に配布し、その登録を依頼します。また、自分のアドレス帳の復号化部分の登録内容を更新します。
もちろん、簡単な暗号化でよい人や、暗号化が必要ない人もいるでしょうが、自分の部屋の鍵を自分で決めるように、自分の意志で、自分の扱う情報の価値にあった暗号化方式を決定できることが重要です。

【００２４】
【機能の拡張】
暗号化、復号化の欄を複数にして複数回の暗号化を重ねることも可能です。特に有料での情報配信では、顧客の決定する暗号方式以外に、最低限の強度を持つ暗号化方式を企業の側で採用し、情報を少なくとも２重に暗号化するべきでしょう。

【００２５】
【電話の場合】
電話でこの方法を利用する場合は、いくつかの通信規格を決定する必要があります。
第１は音声をデジタル化するときのサンプリングレートの規格。
第２はデジタル化した音声を一定量ごとに暗号化する時のその量に関する規格。
この２点が決まれば、電話番号が通知される事を利用して、送信者と受信者の電話番号の順序対によって決まる暗号化による秘密通信が電話の場合でも可能となります。
家庭用の電話機の場合は記憶装置と演算装置を組み込むことは簡単です、電源の問題も心配なく、暗号化と復号化の処理速度の問題だけを解決するだけです。
携帯電話の場合はメモリースティックなどで記憶装置の問題は解決できます。暗号方式を簡単にすれば演算装置の大きさの問題点も解決できます。例えば、データが１０２４ビットごとに暗号化されるとしましょう。暗号化鍵と復号化鍵をこのビット長に相当する文章とし、送信者と受信者で同一の文章を使用します。暗号化方式と復号化方式は XOR とします。暗号化も復号化もデジタルデータと鍵となる文章の１０２４ビットごとの XOR によって実現できます。文章は記憶装置に書き込んでおけばよいのです。XOR だけなら簡単な回路で実現できるので本格的な演算装置を使う必要はありません。

残る問題点として、
第３は記憶装置と演算装置を組み込んだ電話機の制作。
があります。これは新しい需要を喚起する事になります。このようなコンピュータに近い電話機の制作販売は日本の景気回復にも役立つでしょう。携帯電話で使える小型の本格的な演算装置の開発には少し時間がかかるでしょう。

【００２６】
【効果】
この秘密通信の方法は、送信者と受信者の順序対ごとの個別的な形での秘密通信を実現することになります。このときの暗号方式を受信者が自分で決定する事になります。
自分の責任で自分の受信する情報を守ることができるのです。自分が最も信頼する方式を選択できるので、安心してネットワーク社会に参加できるようになります。
安全な情報を受け取れるので、経済価値の高い情報の受信や発信も安心して行うことができます。有料での情報配信の基礎として有効に機能するでしょう。
あくまでも受信者が暗号方式を決定することになるので、自己責任を前提としたネットワーク社会への参加

となり、自立的かつ積極的な姿勢でネットワーク社会に参加するようになります。従って、主体的で活発な国際的交流が可能となるのです。
暗号ソフトの商業的開発や販売がより規模の大きなものとなります。暗号の理論やソフトの作成の面で数学の理論が必要になり数学が役に立つ事を皆が確認できます。日本人の数学の力を情報の安全性向上のために利用でき、それを通じて世界に貢献できるのです。
また、電話機でこの方式を使うことになれば、電話機そのものを新しく制作する事になり、新たな需要を作り出すことになります。この新しい電話機がすべての家庭に取り付けられれば、日本中の家庭がコンピュータで結ばれているのと同じ事になり、新たなサービスの可能性が無限に広がります。

【用語の説明】
１．暗号化部分
平文から暗号文を作るソフトには、暗号化鍵と暗号化ソフトからなるもの、暗号化鍵を使用しないもの、暗号化鍵を暗号化ソフトの中に組み込み一体化させたものなど色々あるが、平文を暗号化する機能を持つ部分を暗号化部分という。
２．復号化部分
暗号文から平文をつくるソフトには、復号化鍵と復号化ソフトからなるもの、復号化鍵を使用しないもの、復号化鍵を復号化ソフトの中に組み込み一体化させたものなど色々あるが、暗号文から平文を復元する機能を持つ部分を復号化部分という。
３．暗号ソフト
　暗号通信に使用する暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の総体を表す言葉である。
４．シーザー暗号
ローマ時代シーザーが使用したと言われる暗号であり、２６を法とする剰余計算を特徴とする。
５．離散対数
Ｚｐ上で、α^e≡a (mod p) の関係があるとき、ｅ をａ　の離散対数と呼ぶ。
６．XOR
排他的論理和を意味する。
計算は、１XOR１＝０，１XOR０＝１，０XOR１＝１，０XOR０＝０となる。
７．復号化
　暗号文からもとの平文を復元させること。

【要約】
【課題】　通信の安全性を強化する。
【解決手段】
（１）送信者と受信者の順序対（S,R）に対して一意的に決まる暗号方式による秘密通信。
（２）暗号化指定欄と復号化指定欄があるアドレス帳を備え、指定欄の内容に従って自動的に暗号化と復号化を行う通信ソフト。
（３）暗号化部分と復号化部分に分割でき、暗号化部分の再配布が自由な暗号ソフト。
【効果】　この１，２，３を利用する方法で秘密通信を実現すれば、暗号解読技術の進歩に合わせて新しい理論を利用した信頼度の高い暗号方式に変えてゆくことができ、秘密を保つ必要がある情報でも盗聴や改竄を恐れずにネットワークで提供できます。情報に安心という付加価値を付けた有料の情報提供をすることができます。暗号化部分だけの再配布を認めることで、商業的に暗号ソフトを開発できます。電話でこの方法を採用すれば、新しい電話機に対する需要を掘り起こすことになります。なによりも、各人が主体的にネットワーク社会に参加するようになるのです。

【おわりに】
　これは特許出願の文書を読みやすくしたものです。特許出願した項目は

　【請求項１】　記憶装置と演算装置を利用できる情報機器による暗号を利用した秘密通信の方法のうちで、送信者（S）と受信者（R）の順序対（S,R）に対して一意的に決まる暗号化方式（Csr）と復号化方式（Psr）に従って暗号を利用した通信を自動的に行うもので、順序対（S,R）に対応する暗号化方式を受信者（R）が前もって自由に指定しておくことができる秘密通信の方法。
　【請求項２】　暗号化の指定欄と復号化の指定欄がある識別記号（メールアドレスや電話番号）一覧を備え、送信時には情報の受信者が前もって指定した方式により自動的に暗号化を行い、受信時には送信者に対応する復号化方式により自動的に復号化することで、請求項１の秘密通信の方法を実現する通信用ソフトウエア。
【請求項３】　暗号ソフトの形式のうち、暗号化部分と復号化部分に分割でき、暗号化部分の自由な再配布を

認めるが、復号化部分の自由な再配布は認めない形式。
　【請求項４】　有料での情報配信の方法で、上記の請求項１，２，３の利用によって少なくとも顧客の要求する以上の強度を持つ暗号化によって情報を配信する方法。
2001 年 1 月 15 日　　宇山靖政

でした。

ただ一つの反響は、斉藤先生からのメールでした。

Subject:　RE: うやまです
Date:　Fri, 11 May 2001 22:26:22 +0900
From:　＊＊＊＊
To:　"Yasumasa Uyama" ＊＊＊

宇山先生

> 　ファックスで送っておきます。
> 時間がとれたら、眺めてみて下さい。

本日、学生の発表練習のため、登校し、
拝読させて頂きました。

先生の基本的な考え方が当方と同じである
事に驚いています。

以下、当方の考え方を述べます。

・離散化された微積分方程式のシステム行列
のモーダル行列をウェーブレット変換行列とし
て使うモーダル・ウェーブレット変換法を考えます。
・境界条件や媒質パラメタを変えた無数のシステム行
列が生成可能です。
・複数個のモーダル・ウェーブレット変換を用いて暗号
表を生成します。
・相手に応じてモーダル・ウェーブレット行列の内積順
番を変更する。または相手のメールアドレス等に応じて
媒質パラメータ変更する。

以上のような Security システムを考えた時期があります。
暗号化テーブルは CMOS メモリーへ格納しておき、長文
の場合、分割して暗号化します。残るは内積演算だけです
から、DSP チップのみが必要です。また、CMOS メモリー
でなく、スマート・メディアを使っても同じです。

先生の先見性に大変驚いて居ります。
Fax を有難う御座いました。

齋藤兆古---------

　出版されても、全く反響はありません。まえにウエーブレットの件で知り合いになった法政大学の教授だけが誉めてくれました。九州大学の教授も見てくださったはずですが、なにも言ってはくれませんでした。雑誌の編集局の方にも、読者からの反響はないとの事でした。
　以前、ＣＱ出版社の雑誌にコンパイラコンパイラの記事を書いたときとは大違いです。そのときは雑誌の読者がとても喜んでくれて、編集局の方へ投稿を寄せてくれました。
　なぜ、反響がないのか。。。

１．つまらないのか？
２．意味がないのか？
と考えて、少し悩みました。この方法は利用者が、世界最新の強力な暗号アルゴリズムを自由に選択し、しかも多重暗号化が可能なので、盗聴防止には有効なのに。
友人に話したら、“盗聴されても気にならない、悪いことはしていないから。”、それなら手紙や電話を盗聴しても良いのか。そう言いたかったです。
通信の秘密は憲法で保証されている。それを守る必要がある状況になった事をあまり気にしていない。経済活動の情報はお金と同じなのに。
でも、私の友人はみんな貧乏人です。お金になる情報は入ってこないし、増やそうとしても元になるお金がない。通信の秘密は必要ない人がほとんどでした。ただ一つ嫌がったのは、携帯のメールや通信が盗聴されたらどうすると尋ねたときだけでした。
盗聴されても気にしない。これぞ、貧乏人の強みですね。“失うものを持たない人は、明るく生きられます。”むかし誰かがそんなことを言っていましたね。

２．６　弁理士さんありがとう

かっこいい名前で請求し直してくれて有り難うございました。特にヨーロッパ、アメリカに出願した名前、Protected Communication System は、とても気に入っています。
何事であれ、実戦は厳しいです。特許申請や特許庁とのやりとりにも経験が必要です。協力してくれる人が必要です。武田信玄の言うように、人は石垣、人は城。ですね。
あれは、夏の暑い日でした。私が東京の弁理士さんの所に行ったのは８月でした。緑の多い田舎に住むものには耐えられない暑さの東京でした。自分は東京には住めない事を感じながら地図を片手に事務所を探しました。最初は事務所とは反対方法に２００メートルほど歩きました。コンビニに入って冷気を浴びながらもう一度地図を確認して反対方向に歩いてやっと事務所に着きました。あのとき出していただいた麦茶はとても美味しかったです。
特許を取るには強力な仲間が必要です。信頼できる弁理士さんを見つけてください。私が弁理士さんの所に行ったとき契約書にサインする事になりました。私が内容を読み始めたとき弁理士さんが言いました。“読んでも分かりません。私を信用してください。”確かにその通りです。信用してサインをして捺印しました。
弁理士さんは、私が貧乏なのを知り、ヨーロッパの代理人とアメリカの代理人に手数料を安くしてくれるように交渉してくれました。おかげさまでとても助かりました。有り難うございます。
何しろ長期戦です。しかも軍資金が無くなったら負けです。お金を払わなければ、特許請求を放棄したことになります。外国は早く結論が出るというのは嘘です。反論を出して勝ち抜くには長い時間がかかります。

２．７　誘惑の手紙（日本の特許紹介業）

日本の特許庁から特許請求の内容が公開されると、すぐに商品化出来るかのような甘い誘いのお手紙が来ます。企業に素晴らしい発明を紹介するパンフレットを作ってくださるとのこと。ありがたくお断りしました。本音を言えば有名人になったようで少し心が揺れました。
注意してください。たとえ公開されても認定されたわけではありません。日本で特許認定されなければ作ったパンフレットはゴミになります。特許認定の結果が出るのはずっと先です。
ゴミを売りつけようとする商売があるのですね。

３．ヨーロッパ出願（２００１年１２月１８日申請受理）

３．１　ヨーロッパに出した請求項：（２００２年０９月２５日公開）

Claims of EP1244267

1. A protected communication computer software applied to an information processing device such as a personal computer or the like having a memory and a communication function over the network, wherein the memory storing a reference table in which a relationship between a transmission destination data and an enciphering program, and a relationship between a transmission source data and a deciphering program being preset, and a plurality of kinds of enciphering programs and deciphering programs, comprising;

means for automatically enciphering a transmission anticipated file according to an enciphering program upon an input of a prescribed enciphering instruction, the enciphering program being automatically specified by looking up the reference table using the designated transmission destination data as a key, and being read out from the memory of the information processing device; and
means for automatically deciphering a received enciphered file according to a deciphering program upon an input of a prescribed deciphering instruction, the deciphering program being automatically specified by looking up the reference table using the transmission source data of a received enciphered file as a key, and being read out from the memory of the information processing device.

2. A protected communication computer software according to Claim 1, further comprising electronic mail transmitting means for transmitting the enciphered transmission anticipated file attached to an electronic mail, to the designated transmission destination.

3. A protected communication computer software according to Claims 1 or 2, wherein;
said reference table is enabled to have a plurality of kinds of enciphering programs and an application order data corresponding to a single transmission destination data, and the transmission anticipated file with transmission destination data designated and correspondence set, is enciphered in multiple stages using a plurality of kinds of enciphering programs in a sequence based on the application order data; and
said reference table is enabled to have a plurality of kinds of deciphering programs and application order data corresponding to a single transmission source data, and an enciphered file specified by the transmission source data with set correspondence, is deciphered using a plurality of kinds of deciphering programs in a sequence based on the application order data.

4. A protected communication computer software according to Claim 3, wherein said reference table is enabled to have a plurality of enciphering programs, or deciphering programs corresponding to each main text of electronic mail and attachment file of electronic mail separately, for each transmission destination data or transmission source data.

5. A protected communication computer software according to Claim 3 or 4, wherein the application order data is automatically changed suitably according to a pre-set rule.

6. A telephone machine having a transmitter notification means such as a caller identification or the like, comprising;
means for installing the enciphering program and deciphering program;
memory means for storing enciphering reference data in which transmission destination data is pre-set in correspondence with an optional enciphering program;
memory means for storing the deciphering reference data in which transmission source data is pre-set in correspondence with an optional deciphering program;
means for enciphering and transmitting a transmission anticipated voice signal using a concerned enciphering program, said enciphering program to be used for the communication being identified based on the transmission destination data regarding the communication, and the enciphering reference data; and,
means for deciphering and producing voice of the received voice signal in order using a concerned deciphering program, said deciphering program to be used being specified based on the data of a transmitter regarding the communication and the deciphering reference data.

7. A telephone machine according to Claim 6, wherein;
said reference table is enabled to have a plurality of kinds of enciphering programs corresponding to a single transmission destination data, and the voice signal transmitted to the transmission destination with correspondence set, is enciphered in multiple stages based on the plurality of kinds of enciphering programs; and
said reference table is enabled to have a plurality of kinds of deciphering programs corresponding to a single transmission source data, and deciphering voice signal transmitted from the transmission source with correspondence set, is deciphered in multiple stages using the plurality of kinds of deciphering programs.

でした。
　請求したらすぐ OK と思っていましたが、そんなに甘くはありませんでした。ヨーロッパ特許庁とのやり

とりについては後ほど詳しく書きます。

３．２　HONG KONG PATENT AGENCY（２００３年０１月２４日）

24th January 2003
We write to advise you that extension to Hong Kong of the following European Patent owned by yourselves must be completed the 24th March 2003　...
という親切なお手紙がイギリスから航空便で届きました。
　気が早いというか、

　誘いの手紙については、ヨーロッパでも同じです。基本的にはヨーロッパの加盟国全体に特許登録できるのです。この手数料を狙って、沢山の国に登録しませんかとのお誘いが来ます。しかも、特許認定されるずっと前から。
　たぶん申し込めばすぐに、手数料の支払いを要求されるでしょう。しかも、ヨーロッパ特許庁が認定しなければ、各国への登録は出来ません。ということは、手数料を払っても、認定されなかったときは何もしてもらえない。業者は手数料丸儲け。どこにもいるのですね。振り込め詐欺のようなものが。
　ヨーロッパ特許庁、アメリカ特許庁に提出したものは請求項の記述をより具体的なものにしました。日本特許庁に提出したものの詳細部分を取り入れて書き換えました。

４．アメリカ出願

４．１　アメリカ出願完了（２００１年１２月１９日）

　ヨーロッパ特許庁、アメリカ特許庁に提出したものは請求項の記述をより具体的なものにしました。日本特許庁に提出したものの詳細部分を取り入れて書き換えました。

アメリカに出した請求項：
Date Code Application Number
Dec 5, 2001 JP 2001-370959
Oct 24, 2001 JP 2001-326908
Dec 21, 2000 JP 2000-388921

Claims
---
1. A protected communication computer software applied to an information processing device such as a personal computer or the like having a memory and a communication function over the network, wherein the memory storing a reference table in which a relationship between a transmission destination data and an enciphering program, and a relationship between a transmission source data and a deciphering program being preset, and a plurality of kinds of enciphering programs and deciphering programs, comprising; means for automatically enciphering a transmission anticipated file according to an enciphering program upon an input of a prescribed enciphering instruction, the enciphering program being automatically specified by looking up the reference table using the designated transmission destination data as a key, and being read out from the memory of the information processing device; and means for automatically deciphering a received enciphered file according to a deciphering program upon an input of a prescribed deciphering instruction, the deciphering program being automatically specified by looking up the reference table using the transmission source data of a received enciphered file as a key, and being read out from the memory of the information processing device.

2. A protected communication computer software according to claim 1, further comprising electronic mail transmitting means for transmitting the enciphered transmission anticipated file attached to an electronic mail, to the designated transmission destination.

3. A protected communication computer software according to claim 1 or 2, wherein; said reference table

is enabled to have a plurality of kinds of enciphering programs and an application order data corresponding to a single transmission destination data, and the transmission anticipated file with transmission destination data designated and correspondence set, is enciphered in multiple stages using a plurality of kinds of enciphering programs in a sequence based on the application order data; and said reference table is enabled to have a plurality of kinds of deciphering programs and application order data corresponding to a single transmission source data, and an enciphered file specified by the transmission source data with set correspondence, is deciphered using a plurality of kinds of deciphering programs in a sequence based on the application order data.

4. A protected communication computer software according to claim 3, wherein said reference table is enabled to have a plurality of enciphering programs, or deciphering programs corresponding to each main text of electronic mail and attachment file of electronic mail separately, for each transmission destination data or transmission source data.

5. A protected communication computer software according to claim 3 or 4, wherein the application order data is automatically changed suitably according to a pre-set rule.

6. A telephone machine having a transmitter notification means such as a caller identification or the like, comprising; means for installing the enciphering program and deciphering program; memory means for storing enciphering reference data in which transmission destination data is pre-set in correspondence with an optional enciphering program; memory means for storing the deciphering reference data in which transmission source data is pre-set in correspondence with an optional deciphering program; means for enciphering and transmitting a transmission anticipated voice signal using a concerned enciphering program, said enciphering program to be used for the communication being identified based on the transmission destination data regarding the communication, and the enciphering reference data; and, means for deciphering and producing voice of the received voice signal in order using a concerned deciphering program, said deciphering program to be used being specified based on the data of a transmitter regarding the communication and the deciphering reference data.

7. A telephone machine according to claim 6, wherein; said reference table is enabled to have a plurality of kinds of enciphering programs corresponding to a single transmission destination data, and the voice signal transmitted to the transmission destination with correspondence set, is enciphered in multiple stages based on the plurality of kinds of enciphering programs; and said reference table is enabled to have a plurality of kinds of deciphering programs corresponding to a single transmission source data, and deciphering voice signal transmitted from the transmission source with correspondence set, is deciphered in multiple stages using the plurality of kinds of deciphering programs.

アメリカ特許庁とのやりとりについては後ほど詳しく書きます。

４．２　逆転する時間（公開中止）（２００２年５月１日）

さすがはアメリカです。時間の流れが外の世界とは異なるようです。

平成１４年（２００２年）４月２２日、弁理士さんから書類が届きました。内容は
２００２年４月１５日にアメリカの代理人が作成した報告書でした。その中に、
２００２年４月３日にアメリカ特許庁が作成した書類が入っていて、
公布予定日：２００２年７月１１日
であると書いてありました。

これを見て、７月には世界中からアクセスがあるに違いない。と思いながらニコニコしていました。しかしその喜びは１ヶ月で消えました。

平成１４年（２００２年）５月２３日、弁理士さんから書類が届きました。内容は
２００２年５月１日にアメリカの代理人が作成した報告書でした。その中に、
２００２年１月１５日の日付でアメリカ特許庁が作成した書類があり、
公布予定日：未定－書類訂正の終了まで

となっていました。

アメリカに出願して驚いたのは、特許公開予定のお知らせが４月作成されているのに、５月になってから、１月の日付の文書で、公開できないとのお知らせが来たことです。

私の特許申請が人を不愉快にさせたのかと心配しました。心の広いアメリカ特許庁が意地悪をするはずがないし、まして正義の国アメリカですから。私の特許は盗聴を防ぐ技術です。正義の国アメリカが卑怯な盗聴なんてするはずはないし、アメリカ政府はプライバシーを守るために盗聴防止技術の発展を応援してくれていると信じています。

表向きの非公開の理由は、ひらがなや漢字が入っているからだそうです。弁理士さんがアメリカの代理人と連絡を取りながら何とか修正してくれました。

インターネットで注目されて世界中の人が見る。と思っていたのですが、現実は厳しいものです。

日本では９ヶ月と少し過ぎると公開されます。きっちり９ヶ月ではありません。アメリカでの公開は無いのかと思いながら、日本とヨーロッパでの公開を待っていました。

きっと、日本中そして世界中から問い合わせが来るに違いない。と思っていましたが、全く問い合わせは有りません。

諦めていたアメリカでの特許申請の公開ですが、ヨーロッパ特許庁から公開されるとすぐに、アメリカ特許庁も公開しました。そして、しばらくの間アメリカ、カナダ、イギリスのヤフーで、Protected Communication System　で検索するとアメリカ特許庁のホームページから公開されているものがすぐに見つかりました。

この１年ほどは検索しても出てこなくなっていました。最近やっと、ヨーロッパ特許庁が特許認定が予定されていることを明らかにしてから、ヨーロッパ特許庁のものがインターネットの検索で見つかるようになりました。アメリカの特許庁のものは検索しても見つかりません。残念です。

それにしても、“何で古い日付の書類が後から届くのだ。”公開されるのか公開されないのかどちらが正しいのか全く分からなくなりました。弁理士さんも困惑していました。弁理士さんからのメールは次のようなものでした。

宇山様、
前略、いつも大変お世話になっております。
さて、米国における特許出願の公開予定日の件ですが、
この度は補正において若干時間が掛かってしまいましたので
当初の予定日である平成１４年７月１１日より遅れてしまうことが
考えられます。
更に、米国特許庁より２通の出願書類受領書が発行されておりますが、
２通目の受領書（平成１４年５月２３日付報告書にて送付済み）にては
公開予定日は補正のために未定となっておりました。従いまして、実際の
公開日は当初の公開予定日よりは遅れてしまうものと思われます。
しかし、幣所にても困惑しておりますのは、２通目の受領書の方が
１通目よりも古い日付になっております。そのため、どちらの受領書に
記されている公開予定日が最新のものなのか分かりかねております。
いずれにしましても、出願が公開されますと、米国特許庁より公開の
旨を通知するレターが送付されます。また、その通知が発行されますと、
現地代理人は公開公報の写しと共に幣所に送付してきますので、届き次第
宇山様に転送いたします。
尚、この度の公開公報には前回行った補正の後にご指摘頂きました
誤記などは修正されておりませんのでご了承下さい。
出願が公開されますと、米国特許庁の公式ＨＰにて公開公報を
見ることが可能になります。その際は下記の url よりアクセスできます。
http://www.uspto.gov/
上記ＨＰより、『Patent』、『Search Published Patent Application』、
『Publication Number Search』
を順番に選択することによって公開番号を基に検索できます。
草々

４．３　勝利の予感（米国公開）（２００２年９月１２日）

何となく邪魔されているのかと思いこんでしまいました。もちろん正義の国アメリカが公布を妨げることなどないのですが、外国には行ったこともなくアメリカの事を良く知らない私は自分の物差しでこの現象を解釈しました。

自分なら都合の悪いものは邪魔して発表させなくする。だから、盗聴できなくなると困る人が公布を妨げようとして変な理屈を付けてきたのだと思いました。

そうならば、これは本当に盗聴を防ぐことのできる技術に違いないと思いこみました。アメリカでは公開されなくても、ヨーロッパ特許庁から公開されれば、それを見た人が作ってくれて、やがて私も誰か他の人が作ったソフトを使わせてもらえるだろうと思っていました。なにしろ、アメリカへの特許申請とヨーロッパへの特許申請の内容はほとんど同じなのですから。

しかし、そこには大きな誤算がありました。どこからも連絡は無いし、誰もソフトを作ってはくれません。それと、携帯電話にそんな機能を付けたら、法律によって輸出出来なくなるなんて、そのときは全く知りませんでした。

もちろん、ヨーロッパ特許庁から公開されてからすぐにアメリカ特許庁からも公開されました。疑ったりしてすみませんでした。

５．企業との対応

５．１　日本の会社（２００３年）

暗号機能付きの携帯電話に関して質問しました。

●質問内容は
安全な携帯を探しています。
貴社で作っている携帯には、
盗聴防止の暗号化機能としては、どのようなものが付けてありますか。

お客様相談センターから回答が来ました。

●回答（１）Date:　　27 Feb 2003 17:21:11 +0900
お問合せの件ですが、盗聴防止機能といたしましては、音声をデジタル
信号に変換して通信を行う処理をしておりますが、これは、弊社独自の
機能（仕様）ではなく、各メーカー共通の機能ですので、ご了承の程、
宜しくお願い申し上げます。

補足質問をしました(2003-02-27 19:26:46)。内容は

●補足質問
デジタル化した音声データを
暗号化したデジタルデータに変換してから、
送信するような携帯電話は作れますか？

●回答（３）Date:　　4 Mar 2003 13:50:35 +0900
さて、携帯電話の補足のご質問につきましては、各通信事業者様の
仕様に基づいた設計となっており、現在のところ、デジタル化した
音声データを暗号化したデジタルデータに変換する携帯電話の開発は
行っておりませんので、ご了承の程、宜しくお願い申し上げます。

そういえば、このころの私は携帯電話を持っていませんでした。携帯電話を持っていない人からの質問を受けてくれる会社はこの会社だけでした。ありがとうございます。

５．２　ノキア（２００３年）

ノキアにも問い合わせました。

●質問
ノキアの携帯には、盗聴防止の暗号化機能としては、どのようなものが付けてありますか。
●回答
この度は、ハローノキアへお問合せ頂きまして誠にありがとうございます。
NOKIA 製品には携帯端末側にて盗聴防止の暗号化機能というものは付いておりません。
音声はネットワークの電波方式によりコード化されておりますので、盗聴はされにく
いものかと思われます。

さらに２回目の質問をしました。

●補足質問
返事をいただき有り難うございます。
質問の追加ですが、
デジタル化された音声データを、暗号化されたデジタルデータに変換してから、
送信する。こんな携帯は
作れますか？
よろしくお願いいたします。

その回答は

●回答
Subject:　お問合せについて
Date:　Mon, 03 Mar 2003 18:00:10 +0900

宇山 靖政 様

ハローノキアへお問合せ頂きまして、誠にありがとうございます。

この度はご提案を頂きまして、誠にありがとうございます。ご存知かとは思います
が、現在の携帯電話の音声は、デジタル化されたデータを細かく分割し、時分割しス
クランブルをかけて転送しております。したがって、その信号を盗聴し、元のデジタ
ルデータに繋ぎ合わせ、さらに音声データにデコードすることは非常に難しく、盗聴
についての安全性は十分確保されていると理解しております。
お客様も良くご存知かと存じますが、これら携帯電話網（ネットワーク）の仕様は携
帯電話網を構築している、各オペレータが担当するエリアで、携帯電話機端末メーカ
は、それらのネットワークの仕様に沿って携帯電話を設計しております。したがっ
て、新たな暗号化などを検討する場合は、携帯電話端末ではなく、ネットワークを含
めた総合的な検討が必要となります。
もし、携帯電話端末に暗号化機能を追加し、同じ機能を持った端末とだけの通信を行
うのであれば、技術的には可能かと存じます。しかしながら、各オペレータより提供
されている携帯電話網に、接続するための機器認定が得られないのではないかと思い
ます。また、機器認定が得られない場合、携帯電話網で使用できない端末になってし
まいます。

上記内容にて甚だ恐縮ではございますが、今後ともノキアを宜しくお願い致します。

さらに、追加の質問をしました。

●補足質問
返事を下さり有り難うございます。
　考えているのは、その
時分割しスクランブルをかけて
ですが、スクランブルの構造を知っている人がいれば

盗聴されてしまうと思います。
私は、
デジタル化されている音声データの
ビット数は変化させないで
うまく暗号化すれば
携帯電話網に接続できて
暗号化も可能になるのではないかと思っているのです。
具体的には、アメリカ特許庁のホームページ
http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=/netahtml/PTO/search-bool.html&r=1&f=G&l=50&co1=AND&d=PG01&s1='Uyama,+Yasumasa'&OS="Uyama,+Yasumasa"&RS="Uyama,+Yasumasa"
（長いのでクリックしただけでは、だめです。URL を全てコピーしてから張り付けて下さい。）
の後半で書いたような
携帯電話が欲しいのです。
こうすれば、同機種で暗号化の登録がしてある時は暗号化機能が働き、
異機種の時は、暗号化機能が無い普通の携帯として
動くと思うのです。（従って、携帯電話網に接続も可能）
いかがでしょうか？

　可能かどうか、検討していただければ幸いです。

－－－－－－－－－－－－－－－－－－－－

　最後の返事は
●回答
Subject:　　お問合せについて
　Date:　　Tue, 04 Mar 2003 17:51:18 +0900

宇山 靖政 様

再度のお問合せをいただき、ありがとうございます。

転送されるデータの内容が制約されていないインターネット上での端末、たとえば IP 電話のような製品では可能かと思いますが、携帯電話のネットワークに接続する携帯電話端末で実現する事は難しいかと思います。
この度は貴重なご提案をいただきまして、誠にありがとうございました。どうぞ今後ともノキア製品をご愛顧くださいますよう宜しくお願い致します。

でした。

　当時は、電話機の暗号化機能には強い制限があるなんて知りませんでした。IP 電話を作ればよいと単純に考えていました。でも、スカイプのようなものがすでに存在し、暗号化機能を備えているようですので、新しく作っても売れそうもないですね。

５．３　ジェトロに相談（２００４年）

　横浜の関内にジェトロがあります。貿易に関して相談を受け付けています。予約をして相談に行きました。まずは相談申し込みから、

//
Subject:
Re: 国際ビジネスよろず相談室申込書 6/11FRI 08:30
Date:
Thu, 17 Jun 2004 15:46:51 +0900
From:
YOK@jetro.go.jp
To:
uyama@kc4.so-net.ne.jp

宇山　靖政　様

このたびは、ジェトロ横浜の国際ビジネスよろず相談室にお申込みいただき、
ありがとうございます。

６月２４日（木）　13：30-14：30 でお時間をおとりしましたので、下記のとおり
ご連絡申し上げます。

****

■相談日時：　６月２４日（木）　13：30-14：30

■場　所：　ジェトロ横浜　（横浜市中区山下町２　産業貿易センタービル　９階）
TEL：　０４５−２２２−３９０１　　FAX：　０４５−６６２−４９８０

//

宇山　様

ご連絡ありがとうございます。

６月２６日（土）１３：３０−１４：３０でしたらお時間を
おとりできると思いますが宇山様のご都合はいかが
でしょうか。ご確認お願いいたします。

宇山様のご都合がよろしければ、２６日（土）担当の
吉川アドバイザーと調整させていただきます。

専門分野が異なりますので宇山様のご相談をお引き
受けできない場合はご了承ください。

ジェトロ横浜
TEL：045-222-3901
FAX：045-662-4980
中熊

//

相談結果は次のような結論でした。

//

宇山様：

6 月 26 日（土）に御相談頂いた内容について、小生の方で再度確認した結果、その場で御説明した通り、
;ソフトは自分で作り、販売は外国の人に頼み、販売手数料を除いた金額を送金してもらう場合、
i 自分の HP からインターネットで直接販売し、購入者から直接、代金を日本の口座に送金してもらう場合、
ともに、輸出規制上、特段の許可が必要ということには当たりません。　従って、基本的に自由に販売することが出来るとお考え頂いて結構です。
尚、代金受領にあたっては、外為法上の手続きが必要となる場合があり、販売に伴う収得金は本邦で課税の対象となることは、御既承の通りです。
今回の御相談がお役に立てば、誠に幸甚です。

JETRO 認定 貿易アドバイザー　吉川泰司

//

しかし、ヤマハのヘリコプターの例もあります。用心用心です。
その後、特許を取得したことを報告したら、次のような返事が来ました。

//

宇山様：

欧州での特許取得の由、御同慶の至りです。　おめでとうございました。
益々の御活躍をお祈り申し上げます。

吉川 泰司

//

その後、アメリカでの特許も取得いたしました。

６．日本特許庁との論争

６．１　日本特許庁の応答

日本の特許庁の判断はいつ頃になるのでしょうか？予測できたらお願いいたします。

と弁理士さんに質問しました。

Subject:　　Re: 日本では？
Date:　　Thu, 23 Sep 2004 15:17:54 +0900
宇山さん、特許庁の審査期間目標は審査請求から２年以内に最初の応答をすることです。

との返事が来ました。公開日が、２００３年７月１８日で、質問した日が２００４年９月２３日ですから、あと１年待てば良さそうです。

６．２　拒絶査定（２００５年６月１６日）

２００５年６月１６日（平成１７年）に拒絶査定の通知が来ました。一番の原因は請求項１でした。

【請求項１】　記憶装置と演算装置を利用できる情報機器による暗号を利用した秘密通信の方法のうちで、送信者（Ｓ）と受信者（Ｒ）の順序対（Ｓ,Ｒ）に対して一意的に決まる暗号化方式（Ｃsr）と復号化方式（Ｐsr）に従って暗号を利用した通信を自動的に行うもので、順序対（Ｓ,Ｒ）に対応する暗号化方式を受信者（Ｒ）が前もって自由に指定しておくことができる秘密通信の方法。

拒絶理由
請求項１には、暗号を利用した秘匿通信の方法なるものが記載されている。ところで、文字、数字、記号などを適当に組み合わせて暗号を作成する方法の発明は、たとえ産業上、殊に商取引において貢献するところが大であり、また作成方法が科学的に精密を極めていても、その間何らの装置を用いないものは、これを暗号による通信方法と解しても暗号による思想表現の方法と認められ、自然法則を利用した技術的手段を施していないから、特許法の対象となる「発明」ではないというべきである。

ようするに、思想表現としてしか請求項１を評価してくれないのです。思想として評価してくれたことを誉め言葉として受け取りました。
私の誤解は、自然法則として、

【利用される自然法則】
２つのもの　Ｘ、Ｙ　に順序を付けて並べる方法は（Ｘ、Ｙ）と（Ｙ、Ｘ）の２種類であり、これ以外にはありません。この自然法則を利用して、順序対方式暗号通信を実現します。

を考えていたのですが、要するに自然法則とは、コンピュータをどう使うかを書きなさいということでした。
私の書いた特許申請の詳細部分を読めば、普通の技術者なら順序対方式暗号メーラーも順序対方式暗号 IP 電話も作成できるのに。詳細部の内容を請求項に書かなくてはならなかったのです。
仕方ないので、新しい思想、２１世紀の暗号通信のあるべき姿を示したと誉めてもらったと考える事にします。

特許法第２９条
特許を受けることができる「発明」であるためには、まず第一に、産業として実施できなければなりません。これは、ただ単に学術的・実験的にしか利用できない発明は「産業の発達」を図るという特許法の目的が達成できず、保護する価値がないからです。
特許法における「産業」は、工業、鉱業、農業などの生産業だけでなく、運輸業などの生産をともなわない産業もふくめた広い意味での産業を意味します。

これから考えれば、社会のお役に立てるのに。
特許庁の人は、日本国民には無料でこの方式を解放してくれ。と言っているのかな。何しろ盗聴法が出来て憲法で保証された通信の秘密が守れない時代だから。しかたない、お国のために協力しましょう。必要があれば（状況を見て）ソースコードも公開しましょう。

そうは言っても、弁理士さんとのやりとりでは何とか特許認定してもらおうとあがきました。

２００５年8月１７日
宇山　様

弁理士の＊＊です。当事務所は８月１４日～８月２１日まで夏休み期間となっておりますが、宇山様の件の拒絶理由通知の応答期限は８月１９日で
すので、自宅にて意見書・補正書の作成を進めております。

宇山様のご用意いただいた反論趣旨は充分に活用させていただきますが、拒絶理由通知との兼ね合いで、いくつかの請求項につきましては、現状では
反論は困難と考えられます。つきましては、理由Ｂが挙げられております現在の請求項１～４は削除せざるを得ないと考えます。この点のご了承をいただ
けますでしょうか。至急、お返事をお待ち申し上げます。

なお、現在の請求項１～４はもともと宇山様の原出願時から存在したもので、私としては、過去の経験からして、特許法第２９条柱書き違反（特許法

第２条に規定する発明に該当しない）を指摘されるのではと、懸念しておりました事項です。

そこで、私としては、その場合の対抗策として、現在の請求項５～１１を提起しておりました。つまり、特許法第２９条柱書き違反は、想定範囲の指摘で
あります。

一方、現在の請求項５～１１は特許法第２９条第２項違反（進歩性なし）ですので、引用文献記載の発明との差異を明確化することで、拒絶理由は
解消する可能性が残されております。とはいえ、宇山様の反論趣旨を考慮しても、勝敗は５０％程度のものではないかと、予想されます。

いずれにしましても、進歩性については、全力を尽くして反論を試みたいと思います。

次のメールは、（２００５年１０月７日）
日本出願の反論ですが、審査段階の結論は出てしまいましたので、次は審判段階に移行せざるをえません。

具体的には、現在の拒絶査定に対して不服申立を特許庁審判部に対して行うことができます。その期限は、１０月２４日となります。

先日、時間稼ぎも含めて、出願を係属中にしておきたい旨も伺っております。不服申立ご希望の場合には、１０月１７日頃までに、その旨をお知らせ下さい。

可能性を検討します。　飯塚

なお、米国の件ですが、送付が遅れておりますが、米国特許庁から審査結果が届いておりますので、今しばらく、お待ち下さい

その次のメールは（２００５年１０月２１日）
宇山さん

日本出願に関する拒絶査定不服審判の請求期限が１０月２４日（月）に迫っておりますが、この件は審判請求いかがしますでしょうか。

私の見たところ、かなり反論は困難ですが、明細書の具体的な記載事項にさらに減縮するほかはないと思います。もっとも、かなり困難な事件ではあります。

もしも、審判請求する場合には、印紙代だけでも８８０００円もかかります。当方の手数料は正規手数料１９万＋頁加算ですが、宇山さんの場合特別料金として全て込みで１０万でお引き受けしま
す。ただし、勝率を考えるとあまりお勧めできません。

もっとも、結論を先延ばしにする目的でしたら、請求項１項に絞って審判請求する等により印紙代は４５０００円程度に減額することは可能です。

至急、審判請求のご意志をお知らせください。

私の返事は、（２００５年１０月２１日）

日本の特許庁への反論はしません。
. .
いま、宣伝用の
無料ソフトを作ったので

そちらで販売の準備をします。
　日本に関しては
他のメーラーで採用しても
文句を言わないことにします。
うやま

でした。

７．ヨーロッパ特許庁

７．１　論争開始（２００４年８月１８日）

　内容をめぐって特許庁との論争が始まります。大量の文献が送られてきました。すぐに反論を書くのです。これが、けっこう疲れます。以下、各特許庁とのやりとりを説明します。偉いお役人様の言い分と平民のひねくれた反論のやりとりです。

　日本特許庁への反論をすることになりました。
　弁理士さんとメールで相談しました。

Subject:　　Re: 書き方
Date:　　Mon, 18 Oct 2004 18:51:08 +0900

宇山靖政様

　前略、いつも大変お世話になっております。
　さて、ご質問頂いた件ですが、審査官は請求項の書き方
について言及しているものであります。

　請求項に関しましては、記載するべき内容と、その表現方法
の２つがございます。

　現在は、宇山様より頂きました資料を検討して、引用文献
との相違点を明確にするためにどの要素を請求項へと
抽出すべきかを検討致している段階でございます。

　宇山様からの案文は、本発明と引用文献との相異点を
明確に理解するという点では大変役に立つものでありますが、
残念ながらその内容をそのまま請求項にすることはできません。
その理由として、請求項とは権利を主張する権利書であり、
独特の書き方が要求されるためであります。
その書き方をお知らせすることはやぶさかではありませんが、
短時間に教えられるものでもありませんし、そこまで宇山様に
お手数をお掛けするのは弊所の本意ではありません。

　従って、宇山様側におきましては、既に頂いております
案文で充分であると考えております。

　本発明や、引用文献との相違点につきまして、これから検討を
進めていく上で、不明瞭な点や疑問点等が生じましたらご連絡
致しますので、その際には助言等を宜しくお願い申し上げます。
　　草々

飯塚国際特許事務所
飯塚

Uyama Yasumasa <uyama@kc4.so-net.ne.jp>さん：
> 宇山です。
>
> 質問があります。
> ５．１の指示に従った書き方ですが
>
>
>
> 序文
> 請求項１
> 請求項の内容
>
>
> となるのか、
>
>
> 請求項１
> 序文（従来技術の特徴と問題点）
> この発明の特徴
>
>
> のように、
>
> 請求項の中を
> ２つに分けるのですか？
>
> よろしくお願いいたします。
>
>
>
>
>
> > 宇山さん、いろいろとご検討いただき、ありがとうございます。さて、
> 実施例の件ですが、日本に関する限り、そのような事実はありません。
> 我々の言う実施例と言う語は、特許申請書類に記載する一具体例のこと
> です。これでしたら、許可の判断に有利に作用することもあります。特
> に、化学の分野の発明では、実施例の記載は必須です。以上
> >
> > --------
> > From: Uyama Yasumasa <uyama@kc4.so-net.ne.jp>
> > Date: 09/30(木)20:54
> > Subject: 実施例
> >
> > うやまです。
> >
> > 日本では、
> > 実施例があると良い
> > というような話を聞きましたが、
> > 来年の夏に返事か来る前に
> > 販売を開始した方が有利でしょうか？
> >
> > よろしくお願いいたします
>
>

以下のように、私なりのヨーロッパ特許庁への反論をまとめて弁理士さんに送りました。（２００４年１０

月１８日）

ヨーロッパへの応答案（２）（宇山の原案）

各文献の指摘された部分に関しての見解

文献＜Ｄ１＞

abstract
A two-phase method of key recovery which will be referred to as Secure Key Recovery (SKR) is presented. The proposed key recovery system permits a portion of the key recovery information to be generated once and then used for multiple encrypted data communications sessions and encrypted file applications. In particular, the portion of the key recovery information that is generated just once is only portion that requires public key encryption operations. We also describe a verification mode in which the communicating parties each produce SKR recovery information independently, without checking the other's so produced information. In this mode, if at least one side is correctly configured, all required recovery information is correctly produced, In addition, the communicating parties are free to include any optional recovery fields without causing a false invalidation of what the other parties sent. Further, we present a method of verification of key recovery information within a key recovery system, based on a variation of the three-party Diffie-Hellman key agreement procedure. Without communication with a trustee, the sender is able to encrypt recovery information in such a way that both the receiver and the respective trustee can decrypt it. This reduces the number of encryptions, and inherently validates the recovery information when the recover decrypts it. The method allows full caching of all public key operations, thus further reducing computational overhead.

要旨

安全な鍵復元（ＳＫＲ）とよばれる二相式の鍵復元方式について提案します。提案された鍵復元システムは、一度だけ作られた鍵復元情報の一部分を、多くの暗号化されたデータによる情報交換、会話、暗号化されたファイルのアプリケーションに利用できます。

とくに、ただ一度だけ作られる鍵復元情報の一部分だけが、公開鍵による暗号化を必要とするただ一つの部分です。

検証モードについて説明すれば、そこでは、互いに情報交換をする２集団はそれぞれの SKR 復元情報を、他が作成した情報に関与することなく、独立して作成します。

このモードでは、もし、少なくとも一方が正しく環境設定すれば、全ての復元情報は、正しく作成できます。

さらに、情報交換をしようとする集団にとって、復元情報部分を加える必要は全く無い。他の集団が送ったものを誤って無効にすることはない。

さらに、私たちは鍵を復元するための情報を、Diffie-Hellman 鍵同意過程を３集団の場合の変形したものに基づいて、鍵復元システム内で確認する方法を提供する。

受託者との情報交換なしで、、送信者は復元情報を、受信者とそれぞれの受託者が復号化できる形で暗号化することが可能である。

これは暗号化の数を減少させる、また受信者が復号化するときに、復元情報が、内容的な正しさを与えてくれる。

この方法は、全ての公開鍵による操作を完全にメモリー上で行うことを可能にし、さらに、計算によるの負荷を軽減する。

＜反論＞

問題にすべきは、暗号の利用者が、自分の必要とする強度を実現する方法と、ネットワークに関する権限や、国際間の規約としてのアルゴリズムに対する ID を決めたりする権限を持たない人でも可能な範囲内で利用できる現実的な方法を提起することです。

さらに、３年もすれば解読方法が発表される暗号の世界の中で新しい暗号がすぐに利用できる環境、新しいアルゴリズムを使用した暗号ソフトが販売しやすくなる形式を発見することが大切です。誰もが簡単に利用できる暗号ソフトの形式が必要です。しかも、制作者の経済的な保証が出来るようにしなくてはなりません。

最後に、アルゴリズム開発の遅れをカバーする方法が必要です。これらが揃えば、利用者は安心して重要な情報をインターネットに流せます。

犯罪の取り締まりには、令状をとって、必要なコンピュータを押収すればよい。

１．Tow-phase Cryptographic Key Recovery System
p. 483 - 485
Key Recovery Environment
鍵復元環境

The key recovery system described in this paper will be referred to as Secure Key Recovery (SKR). A key for which SKR key recovery information has been created will be called an SKR-protected key. An SKR-protected key can be any type of secret key, although the discussion will be limited primary to data keys. Data keys are keys used to encrypt messages transmitted within sessions or within E-mail applications or keys used to encrypt data files. A policy option provides that a user may reveal only a part of a given SKR-protected key.

この文書で述べられる鍵復元機構を安全な鍵復元（SKR）と呼ぶことにする。ある鍵に対して、SKR 鍵復元情報が生成されているとき、その鍵を SKR 保護付き鍵とよぶ。SKR 保護付き鍵は秘密鍵の一種であるが、ここでの議論はデータ鍵に関してのみなされます。データ鍵は、情報交換の中で暗号化されたメッセージが伝えられる場合や、電子メールのアプリケーションや、データファイルを暗号化する鍵です。提起された方針は、各ユーザーが与えられた SKR 保護付き鍵の情報の一部を明らかにすればよいということです。

＜反論＞
しかし、鍵の情報をその一部であれ明らかにすることは暗号化の意味をなさない。最強の暗号化を施すことをその目的にすべきである。自分から鍵情報を外部に漏らすことは避けるべきである。

Communications can be among two or more parties, although the discussion will be limited primary to two-party communications. The SKR protocol can be easily extended to three or more parties. The SKR protocol supports session communications, wherein a sender and receiver interact in real-time, and store-and-forward communications such as electronic mail and file transfer applications, wherein the sender and receiver do not interact in real-time. Also, the SKR protocol supports file applications involving a single user or application. In this latter case, the application or user who creates a data file is the same application or user who creates a data file is the same application or user who retrieves the data file.

議論は主に２つの集団間の情報伝達に制限されるが、２つまたはそれ以上の集団間の情報伝達に関しても適用できる。SKR 通信規約は３またはそれよりも多数の集団間での情報伝達の方法にも拡張できる。SKR 通信規約はいろいろな形式の情報伝達の場面で利用できます。たとえば送信者と受信者が同時に情報を交換しあう場合や、保存ー転送を基本とする電子メールやファイル転送のような場合にも利用できます。

Figure 1 shows a communication system in which SKR can used. A sender (Alice) in country X communicates which a receiver (Bob) in country Y by transmitting one or more encrypted messages (making up a communications session or electronic mail application) or transmitting one or more encrypted files (making up a file transfer application) over a communication channel. The sender and receiver may each comprise computer workstations, suitably programmed to provide the encryption and key recovery functions described below. Although the example of Figure 1 assumes that the sender and receiver are located in different countries (X and Y), the scheme may be used entirely within a single country.

図１は、SKR が適応できる通信機構を表しています。送信者（アリス）はＸ国にいて、Ｙ国に住む受信者（ボブ）と一つ又はそれ以上の暗号化されたメッセージを送信することで、情報交換をします。（同時情報交換や、電子メール用のアプリケーションで情報交換をしているとします）または、一つ又はそれ以上の数の暗号化されたファイルの（ファイル転送アプリケーションを使って）伝達経路を使っての転送を表しています。送信者と受信者はそれぞれが、暗号化機能と鍵復元機能をみたすようにプログラムされたワークステーション型のコンピュータを持っているとします。
この図１では、送信者と受信者はそれぞれ異なる国（Ｘ国とＹ国）に住んでいるとされていますが、この計画は両者が単一の国に住んでいる場合にも適用できます。

The example of Figure 1 specifically addresses the case where there are two different communicating parties (Alice and Bob). However, this two-party model also describes (to a degree) the one-party model in which Alice encrypts files. In that case, the file encryption application may be thought of as a store-and-forward application, where Bob=Alice and a long time exists between the file sending (Alice acting as the sender) and file receiving (Alice acting as the receiver) operations. However, obvious differences affecting the protocols in each of these cases will be pointed out.

図１の例は、特に、２つの異なる集団（アリスとボブ）がある場合のことをあらわしている。この２集団を使った具体例は、１集団の場合すなわちアリスがファイルを暗号化するというようなときでもそのまま利用できる。その場合は、ファイルを暗号化するアプリケーションを保存ー転送型のアプリケーションと考えます。ボブ＝アリスと考え、ファイルの送信（アリスが送信者として振る舞う）と復元（アリスが受信者として振る舞う）の間に長い時間がかかる操作だと言うことができます。この場合に規約を適用するときの明らかな違いは指摘しておく必要がある

The transmitted messages are encrypted by the sender using a data encryption key and decrypted by the receiver using corresponding data decryption key. If a symmetric encryption scheme (e.g. DES) is used, then the data encryption key (which is also the data decryption key) is the 'secret' data key for the purpose of key recovery (i.e. the SKR-protected data key).

送信者によって暗号化されて伝えられたメッセージは、受信者によって対応するデータ暗号化鍵をつかって復号化されるとします。もし、(DES のような)対称的な暗号技法が使われるときは、データ暗号化鍵（これが同時にデータ復号化鍵でもある）は（SKR 保護付きデータ鍵のような）鍵復元のための“秘密”データ鍵になる。

However, the SKR system is not limited to the protection of data keys. The SKR system can also be used to provide key recovery services for any secret key. For example, if an asymmetric encryption scheme (e.g. RSA) is used then the private part of the key is the 'secret' SKR-protected key for the purposes of key recovery.

SKR 機構はデータ鍵の保護だけに制限されるものではない。SKR 機構は任意の秘密鍵に対する鍵復元サービスにも使える。例えば、(RSA のような）非対称鍵を使う技法に対しても利用できる。そのときは、鍵の私的部分が、鍵復元のための、“秘密”SKR 保護付き鍵となる。

A part (in this particular example) of key recovery agents are selected in country X, while a part of key recovery agent are selected in country Y. The establishment of key recovery agents could take place as part of the establishment of a general public-key infrastructure. Each key recovery agent has at least one public and private pair of keys (denoted PU, PR). The public and private parts of the key for Alice's two key recovery agents in country X are (PUx1, PRx1) and (PUx2, PRx2) and for Bob's two key recovery agents in country Y are (PUy1, PRy1) and (PUy2, PRy2). This enables Alice and Bob to use public key cryptography (e.g. RSA, Diffie-Hellman, or Elliptic Curve algorithms) to encrypt and protect certain SKR information.

この特別な例では、X 国において、一組の鍵復元窓口が選ばれ、それに対応してＹ国でも一組の鍵復元窓口が選ばれる。鍵復元窓口は、一般的な公開鍵のための社会基盤の一部として作ればよい。
それぞれの鍵復元窓口は少なくとも一つの公開鍵と私的鍵の組を持っている。(PU,PR と書くことにします）Ｘ国におけるアリスの鍵の２つの鍵復元窓口での公開部分と私的部分の組はそれぞれ(PUx1, PRx1）と(PUx2, PRx2)です。Ｙ国におけるボブの２つの鍵復元窓口における公開部分と私的部分は、それぞれ（PUy1, PRy1）と（PUy2, PRy2)です。これによって、アリスとボブは暗号化したり、SKR 情報を守ったりするのに、公開鍵暗号（RSA,Diffie-Hellman,楕円曲線暗号）方式を使えることになります。

＜反論＞
　暗号化の社会的基盤として、公開鍵を集めた場所が考えられるが、公開鍵を公表することは鍵の長さが分かり、解読されやすくなる。普通は、鍵を公開しなくても少しの知人にだけ知らせておけば十分である。しかも、それぞれに対して異なる公開鍵を与えておくのがよい。さらに、現在の時点で鍵復元窓口は存在するのであろうか？
　この観点から、この暗号化は暗号を弱いものにしてしまう。しかもすぐに利用することが出来ない。その実現には世界各国の同意が必要となる。利害の対立する国がこのような暗号の利用方法に同意するとはおもえない。その意味でも実現するのが無理な方法である。

Communications are assumed to be subject to interception by third parties, including law enforcement agent in countries X and/or Y. Non-governmental third parties intercepting the encrypted communications will be unable to decipher the communications unless they successfully use one or more cryptanalytic techniques. On the other hand, a law enforcement agent having proper authority will be able to recover an SKR-protected key, using the key recovery agents for his country as described below, and the sender (Alice) or receiver (Bob) will also be able to recover an SKR-protected key using a similar procedure.

情報交換の様子は X と（または）Y 国における法の執行者を含む第３者によって傍受される状況下にあると仮定する。伝達情報を傍受する、政府ではない第３者は、一つまたはそれ以上の暗号解読技術を成功裏に適用することなしには、暗号化された情報伝達を解読することは出来ないでしょう。その一方で、SKR 保護付き鍵についての正当な権利を持っている法の執行者は以下に示すようにして、彼の国での鍵復元窓口を利用してもちろん、送信者アリスと受信者ボブもまた同様の過程を経て、SKR 保護付き鍵を復元できます。

＜反論＞
　鍵復元窓口へのハッキングが盗聴を企てる第３者が最初にすることであろう。蓄積された情報は高い値段で売れるので、鍵復元窓口の内部から密かに持ち出され販売されやすい。蓄積されたクレジットカード情報の流出の事件は多い。このような施設は、逆に第３の盗聴者のお手伝いをする危険性が大きい。
　秘密は分散させておくべきである。これが、情報の安全性のために必要な原則である。これに反して鍵を集積することは自殺行為である。通信を傍受する正当な権利をそのときは持っていた人間もいつかはその立場が変わり、正当な権利を持たなくなっても記憶は残る。過去の適法に入手した情報を法に違反する形で使用する可能性があるのだから。

Two-Phase Protocol
２段階式の手続き

The SKR system call for a two-phase process of embedding key recovery information within encrypted communications or encrypted files, consistent with the law and regulations of the country or countries where the encrypted communications take place or encrypted files are produced.

SKR 機構は、２相式の手順で、鍵復元情報を暗号化された伝達情報や暗号化されたファイルの中に埋め込んで、法律、国や他の国々の規制と調和した、暗号化された伝達情報や暗号化されたファイルを作成します。

Figure 2 shows the two phases of the SKR Communication Process. In Phase 1, the communicating parties (Alice and Bob) establish a common random seed (S), from which random key-generating keys (KG) specific to each recovery agent (not shown in Figure 2) are derived. The KGs are encrypted using the public keys of the respective agents. The encrypted KGs are included in an SKR Block 1(B1) prepared by Alice, which she sends to Bob.

図２は、２段階の SKR 情報伝達過程を表している。第一段階では情報交換をする（アリスとボブ）集団間で

共通の乱数の種数（Ｓ）を設定し、それからランダムな過程を経て鍵（KG）を生成する。このかぎ（ＫＧ）は特別に、それぞれの鍵復元窓口に引き継がれる。このかぎ（KG）は、鍵復元窓口の持つ公開鍵によって暗号化される。暗号化された KG はアリスによって用意された SKR ブロック（Ｂ１）に含まれる。そしてボブに送られる。

The manner in which the random seed (S) is established depends on the communications environment. In an interactive session, Alice and Bob could establish S interactively (e.g. using a Diffir-Hellman key exchange protocol), or Alice could generate S and send it to Bob (e.g. using an RSA key transport protocol). In store-and-forward applications, Alice would send S to Bob (as mentioned above), although it would be possible for Alice and Bob to set up a special session, allowing them to establish S interactively.

乱数の種数（Ｓ）を設定する方法は情報伝達するときの環境に依存する。アリスとボブは（Diffir-Hellman の鍵交換手続きを使ったりして）お互いに鍵交換を果たす。または、アリスが鍵を作成し、（RSA 鍵転送手順に従って）ボブに送ることも出来る。前に書いたように、保存転送アプリケーションを使ってアリスがボブに鍵を送ることが出来る。また、アリスとボブがお互いに特別に集まって、彼らのＳを制定することもできる。

In Phase 2, the communicating parties (Alice and Bob) establish a secret data key K. Any key transport or key agreement mechanism can be used to establish K, as the means by which K is established is independent of the SKR system. The data key (K) is encrypted in a nested fashion using a symmetric algorithm and the key-encrypting keys (KK) derived from the stored KGs (via a hierarchical key derivation procedure). If allowed, only part of the data key K is made available in the recovery information, by holding back a portion, R, of the encrypted K. The encrypted K, or portion thereof (if a portion R is related), is included in an SKR Block 2 (B2) prepared by Alice, which she sends to Bob. For file encryption applications, where Bob = Alice, Alice would merely generate and store K, and the encrypted K would be made available by including it in an SKR Block 2 prepared and stored by Alice, until a later time when it may be needed by the recovery process (i.e. it is sent from Alice to Alice with a long time delay). Finally, the data encrypted with K is communicated or stored in a file.

第２相では、相互通信をする集団（アリスとボブ）は秘密データ鍵 K を設定する。鍵送信と鍵同意する仕組みは鍵Ｋを設定するのに利用される。Ｋの設定は SKR 機構とは独立している。データ鍵（K）は、入れ子状に暗号化される。対称鍵アルゴリズムと保存されている KGs から生成された、鍵暗号化のための鍵（ＫＫ）を使って暗号化される。（すなわち、階層的な鍵生成過程）である。
もし許されるなら、データ鍵の一部分、暗号化された鍵Ｋの後半部分Ｒだけが復元情報として利用される。暗号化されたＫ、またはその一部（Ｒの部分が関連づけられている場合）がアリスによって作られるＳＫＲブロック２（Ｂ２）に含まれる。これをボブに送る。
ファイル暗号化のアプリケーションでは、ボブ＝アリスであり、アリスはＫを生成してそれを保存するだけです。アリスによって準備され、保存された SKR ブロック２に含まれる暗号化されたＫは、後ほどまで保存され、復元過程で必要となる。（すなわちアリスから、長時間の遅れを伴ってアリスに送られる）。最後に、Ｋによって暗号化されたデータが伝達されるか、ファイルとして保存されることになる。

The division of the key recovery preparation process into two phases allows the separation of the ‘Establish S’ operation and the associated public key encryptions required in the preparation of ‘SKR Block 1’ (Phase 1) from the per session exchange of K and the associated calculation and verification of the session key recovery information (Phase 2). The information provided in Phase 1 (i.e. S and SKR Block 1) is not in any way based on the per communication or per file information, and can therefore be used across multiple communications or created files. If the information (S and SKR Block 1) is cached, the expensive public-key encryptions done in Phase 1 can be avoided in all but the first communication between parties or first created file, thus greatly reducing the computational effort needed.

鍵復元の準備過程が２段階に分かれることから、Ｓを制定する過程と SKR ブロック１を作成する（第１段階）を、Ｋの交換と関連する計算とセッション鍵の復元情報（第２段階）の検証から分離できる。
第一段階で提供された情報（すなわち、ＳとSKR ブロック１）は、各コミュニケーションや各ファイル情

報を基礎にしたものではない。それゆえ、多くのコミュニケーションや作成されたファイルに対して使えるのです。もし、（ＳとＳＫＲブロック１）の情報が洩れたとしても、第１段階でなされた公開鍵による高価な暗号化は、２つの集団間の最初の情報交換と最初に作成されたファイルを除いて、全ての計算能力、計算量の必要度を大きく下げる。

Figure 3 shows the SKR key recovery process that allows SKR-protected keys to be recovered. A Recovery Service interacts with Key Recovery Agents 1 and 2 to recover a key K. The Recovery Service may be operated by a independent Service Provider party on behalf of users or companied, who want to recover damaged or lost leys, or it may be operated by law enforcement on behalf of law enforcement, who want to recover keys in order to read intercepted communications or files of individuals known or suspected of engaging in criminal activities.

図３では、SKR 鍵復元過程は、SKR 保護付き鍵が復元されることを認める。復元サービスでは鍵復元窓口１と２との情報交換によりかぎＫを復元する。

サービス供給集団とは独立して、ユーザーや会社の立場に立って復元サービスは行われ、壊れたまたは失った鍵を復元する。またそれは、法を執行する時には、法の執行の味方をし、犯罪行為をするであろうと疑われたり、犯罪行為をすることがよく知られている場合や、犯罪行為をすることがほぼ確実な場合には、傍受した通信内容や、ファイルを読むために鍵を復元する。

＜反論＞

暗号化された通信を傍受したものが、犯罪に関係がないかどうかは内容を見なければ分からない、人が最初に犯罪を犯す前は全く善良な市民であり、何時犯罪者になるかは分からない。犯罪に関係ないという確証は通信内容を見た後で決まることであり、善良な市民が犯罪者になるのだから、通信内容が犯罪者になる瞬間の内容を含んでいるかもしれないので、全ての通信内容を確認しなくてはならない。なにしろ、生まれながらにして犯罪者という人間はいないのだから。

疑わしいのは全ての善良な市民であり、疑わしいのは全ての通信である。もちろん犯罪者として有名になったひとたちの通信は疑わしいのではなく、前もって犯罪的なものだと予見できるが、その人たちの通信内容が全て犯罪に関係しているわけではない。見た後で過去に罪を犯した人の通信が、犯罪に関係ない内容の通信を行ったことが分かる時もあるが、知ってしまった内容は元には戻せません。

最近は、政治家が闇献金で検挙されることも多い。彼らを犯罪者予備軍として通信傍受の対象にしてその通信内容を傍受し国民全体に全ての内容をインターネットで公開すべきであろう。なぜなら、一部の人だけが知っている情報は価値を持ち販売可能であり、法律に違反して内部から流出し販売されることがあるからであり、新たな犯罪を防ぐには、公開するしかない。そうしないと、通信を傍受し解読した人が新たな犯罪者予備軍として通信傍受の対象になるからです。

失った鍵が復元できるなら、内部流出した１，２の情報、またはハッキングされた１，２の情報から、公開鍵暗号での私的鍵や、共通鍵方式の秘密鍵が復元されてしまう。真にユーザーや会社の味方なら、私的鍵や共通鍵をいっさい外部に出さないようにする方法を提起すべきである。

鍵を無くしたら別の鍵のセットを作成し公開鍵だけを相手に送信し再度暗号化したものを送ってもらえば良い。企業の秘密情報を解く鍵を他人に預けることは、大切な情報がありますので、ハッキングしてくださいと言っているようなものです。暗号化したことや鍵そのものの存在も秘密にするべきです。また、鍵を分割し、そのうちの幾つかだけが揃えば復号化できる方式もあるので、組織内部の責任者数人で分割保存し、全員が揃わなくても解読できるような方式にすべきである。決して組織以外にものに知らせてはならない。

A party who want to recover a key first provides the needed SKR Blocks (1 and 2) to the Recovery Service. Recovery Service then provides SKR Block 1 and part of SKR Block 2 to Key Recovery Agent 1. In turn, Key Recovery Agent 1 uses the provided information together with the private part of his key to decrypt the encrypted KG value, and then to calculate a key-encrypting key (KK) from KG, which is returned to the Recovery Service in the “Decrypted Information”. Likewise the Recovery Service provides SKR Blocks 1 and part of 2 to key Recovery Agent 2. In turn, Key Recovery Agent 2 uses the provided information together with the private part of his key to decrypted KG value, and then to calculate a key-encrypting key (KK) from KG, which is returned to the Recovery Service in the Decrypted Information. The nested encrypted value of K in Block 2 is withheld from the key Recovery Agents. In that way, the Recovery Agents themselves cannot recover K. The two key-encrypting keys are then used by the Recovery Service to

decrypt the encrypted K in SKR Block 2, and the clear value of K is returned to the requesting party.

鍵を復元したいと思った集団は、鍵復元機構に SKR ブロック（１と２）を鍵復元サービスに提出します。鍵復元サービスは SKR ブロック１と SKR ブロック２の一部分の情報を鍵復元窓口１に提出します。
それに対して、鍵復元窓口１は与えられた情報を彼の鍵の私的部分と一緒にして、暗号化された値 KG を復号化します。そして、KG から、鍵を暗号化するための鍵(KK)を計算します。この復号化された値が復元機構に戻されます。同様に、鍵復元サービスは SKR ブロック１と２の一部分を鍵復元窓口２に提供します。
これに対して、鍵復元窓口２は、その私的な情報と自分の持つ私的鍵を使ってＫＧの値を復号化します。そして、鍵暗号化の鍵 KK をＫＧから計算します。それが、復号化された情報として復元サービスに返されます。
その、入れ子状に暗号化された鍵ブロック２におけるＫの値は鍵復元窓口２からは隔離されています。このようにして、鍵復元窓口だけではＫを復元できません。二つの鍵暗号化のための鍵は復元サービスによって、SKR ブロック２によって暗号化された鍵Ｋを復号化するのに使用されます。そして、完全なＫの値が要求した集団に返されます。

Law Enforcement can obtain a key K only if it has a valid warrant or court order that can be provided by the Recovery Service to Key Recovery Agents 1 and 2. Likewise, a user or company can only recover K if it provides the correct authentication information to the Recovery Service, who in turn provides the authentication information to each key authentication information to validate the user's or company's right to recover K.

裁判所の命令や正当な令状に基づき、これらが、鍵復元サービスから、鍵鍵復元窓口１と２に対して提出する形で法を執行すれば、Ｋを得ることが出来ます。
同様に、利用者自身や会社なども正当な理由を示す証拠を復元サービス機関に示せばＫを復元できます。逆にこの機関は、個人または企業がＫを復元する正しい権利を持つことの認証情報を提供します。
それぞれの鍵の認証情報を提供します。

＜反論＞
現代の犯罪では、蓄積された個人情報が一挙に大量に流出し違法な形で販売されます。復元窓口１と２に蓄積された情報か違法に流出し大量のＫが復元されてしまうことが心配されます。情報流出は外部からのハッキングだけではなく内部からの情報の持ち出しによって起こることも多く見受けられます。特に鍵Ｋが共通鍵方式の場合は致命的となります。
鍵の情報は、たとえ盗まれても社会全体に及ぼす被害が少ない形で保存されなくてななりません。この点、順序対を基本とした私の考えが大切になります。アリスのコンピュータがハッキングされ蓄積された情報が洩れても、アリスが送信した内容を復元するための鍵はアリスのコンピュータにはありません。それらの鍵は送信相手のコンピュータの中にあります。
アリスの送信する情報を解読するには、送信相手全てのコンピュータの内容を盗み出さなくてはなりません。これは、送信相手のコンピュータに少なくとも電源が入っていない限り不可能です。復元窓口のコンピュータは常に電源が入っているので、ハッキングの対象になりやすい。何しろある国の国防省のコンピュータに進入するものもいるのだから。
最大の防御は電源を切ることです。個人のコンピュータなら電源を切れます。公的なコンピュータは電源を切れません。防御すべき情報を集積し、ハッカーに攻撃目標を与えるよな方法は愚劣な方法です。秘密は完全に分散させるべきです。
さらに、私の方法では、鍵ファイルが存在しないときは本人が直接キーボードからパスワードを入力するようにしたり、フロッピーディスクを挿入するように要求させることも出来ます。この場合は電源が入っていても鍵はコンピュータの中には存在しませんので、特に安全です。

p489
SKR Design Details
SKR 設計の詳細

SKR is designed such a way to allow sufficient flexibility and options for cases involving interactive or non-interactive communications. The following subsections discuss SKR design

details.
SKR は十分な柔軟性を備えた形で設計されている。対話方式にも非対話的な情報交換にも対応できる。以下の節は SKR 設計の詳細を説明してあります。

Selection of Encryption Algorithms Within SKR.
SKR における暗号化アルゴリズムの選択

SKR is independent of the session-level encryption used between Alice and Bob. SKR itself, however, uses both symmetric and asymmetric encryption to calculate the values within the SKR blocks. The encryption algorithms used within SKR may also be freely chosen, and the choices are indicated as parameters within the T1 and T2 headers. By default, RSA is used for public key encryption, and a keyed shuffler (described below) is used for symmetric encryption. The keyed shuffler is used by default, as it has the useful property of accepting arbitrary size encryption keys, and arbitrary size data blocks. Another possible choice for the symmetric encryption used within SKR would be to use the same encryption algorithm as used by the application for the session level encryption. This would offer the advantage of availability, as the application has already implemented it, and therefore the symmetric key encryption with SKR would have the same strength as the encryption used to protect the session data.

SKR はアリスとボブの間での暗号化の方式とは独立である。SKR それ自身は、対称または非対称の暗号化方式の両方を使用して、SKR ブロックの値を計算する。SKR ないで使われる暗号化アルゴリズムは自由に選べる。その選択結果は、パラメーターで表され、先頭部分にあるＴ１とＴ２内に記述される。特に指定しなければ、公開鍵暗号化としては RSA が使われる。鍵をかけた混合過程では（以下に述べるように）対称鍵による暗号化が使われる。指定がなければ鍵をかけた混合過程では、どんな長さの鍵でも受け付けるという性質を持っている。任意の長さのデータブロックにも対応する。
他の可能な選択は、SKR において使われる共通鍵方式の暗号化に、情報交換する時に使う時の暗号化と同じ暗号化アルゴリズムを使うことである。
これを使うと、アプリケーションすでに確立されているとき、交換される情報が SKR での対称鍵暗号化と同じ強さで保護されるという優位点がある。

＜問題点＞
T1,T2 のパラメータで、暗号化アルゴリズムを表すことは、アルゴリズムの使用に関しての自由度を失わせる。新しい暗号アルゴリズムを発明した人が、暗号ソフトを売り出すとき、この新しいアルゴリズムを表すパラメータは、誰が、何時、どの様にして決定するのであろうか？制作者が自由に決めると重複する事が起こる。新しいアルゴリズムは、３年もすれば廃れたアルゴリズムになる。解読技術が発達し、コンピュータの計算速度が増すからである。発表と同時にアルゴリズムのパラメータが決められない方式では意味が無く、国と国の対立を考えると、すぐにパラメーターが決まるとも思えない。
つぎに、データの安全な伝達のためには、異なるアルゴリズムで何重にも暗号化すべきである。ヘッダー内にパラメータを置くと、ヘッダーサイズに制限があり、多重暗号化に強い制限が加わる。
どんな長さの鍵でも使えるのは当然のことである。同時にどんなアルゴリズムもパラメータ無しですぐに使えるような形を考えるべきである。

p498

Global Jurisdiction/Policy Table
世界的　司法／政治　表

Referring to Table 1, the information required by the key recovery system of the SKR system is stored in a table called the global jurisdiction/policy table. Table 1 is for illustrative purpose only. In an actual implementation the data would be stored appropriately, perhaps in separate tables, one specifying the key recovery agents' public keys and one specifying the system to calculate the sizes of the keys and key recovery values for specific algorithms and users located in different countries. It may also contain the public keys of recovery agents authorized for each country. The numbers in the table are examples only to demonstrate the kind of flexibility the SKR system permits. The variations are virtually unlimited. In particular, each country may have many key recovery agents.

表１に関しては、SKR システムの鍵復元システムで必要となる情報が保存されたものです。
この表は、世界的な司法／政治表と呼ばれます。表１は説明のためにのみ作成されたものです。実際に構築するときは、データは鍵復元窓口の公開鍵の表と、規則の表に分割されているような形で適切に保存されるでしょう。これは、ある鍵復元窓口における特別な場合を示したものである。このテーブルからシステムは異なる国に住む利用者が使う、特定の暗号化アルゴリズムに関して、鍵復元の値と、鍵の長さを計算する。
ここには、各国で認定されたところの鍵復元窓口の持つ公開鍵も含まれる。この表の中の数値は、SKR システムが持つ柔軟性を強調するための具体例にすぎない。変更は無制限である。とくに、各国では多くの種類の鍵復元窓口を設けるかもしれない。

＜問題点＞
この表には、大きな問題点がある。
１．暗号化アルゴリズムの項はあるが、復号化アルゴリズムの項はない。
２．国の項がある。
３．宛先であるメールアドレスの項がない。
４．鍵の長さ、暗号化アルゴリズムに制限がある。
５．この表の置き場所はどこか。
以下、各問題点に関して検討します。
１．について、
暗号化アルゴリズムの欄にある幾つかのアルゴリズムに ID を付けて、その ID によって、アルゴリズムを判別し復号化する。と主張するのであるが、これでは新しい暗号化アルゴリズムをすぐに使うことが出来ない。なぜなら、アルゴリズムに ID を付けないと利用できないからである。
暗号の種類は沢山あるが、自分の気に入ったアルゴリズムが事前に登録されていて、その ID が決まっているかどうかは分からない。たとえ決まっていたとしても、様々な暗号アルゴリズムの全てが登録されているとしても、逆に困ることがある。アルゴリズムの数が多いと記入する ID を間違える可能性がある。
過去に発表された全ての暗号アルゴリズムに ID を付けて登録することは困難である。新しいアルゴリズムには ID が付いていない。この状態が出現する。問題が多い方法である。
ある人にとって、気に入った暗号化アルゴリズムは１０個くらいであろう、この１０個を自分で ID 無しで管理するのは簡単である。
新しい暗号アルゴリズムを発表されたらすぐに使うようにするには、ID は使えない。暗号化して送信するときと、受け取った暗号文を復号化するときでは、異なるアルゴリズムを使うのが解読困難にする方法の一つである。ID 無しで二つのアルゴリズムを使うには、暗号化ソフトの項と、復号化ソフトの項が必要となる。もちろんそれに伴う暗号化鍵の項と、復号化鍵の項も必要となる。

２．について
国の ID は、どの様にして決まるのであろうか？新しい国が生まれてその国の ID が決まるまではメールを送信できないのであろうか？古い国 ID で送れるのだろうか？分裂前の国では鍵の長さ無制限で、新しい国は暗号化を許さないときは、この ID がすぐに決まり、法律を正しくしかもすぐに反映しなくてはならない。
この仕事は誰がやるのか、そのコストは誰が負担するのか？　もし、市や町の条例で暗号規制をした場合、市や町ごとの表が必要になる。こんな表は管理しきれない。その意味で現実性がない。

３．について
国を宛先と考えることは出来ない。暗号技術の専門家から見たら暗号とはいえないものも多い。国ごとに利用できるアルゴリズムの表を作成して ID を付けると、この場合価値のないアルゴリズムだけが表の中に載っている場合も考えられる。
メールの宛先を記入しなくてはならない、暗号メールを受け取る専門家にとっても意味のあるアルゴリズムの一覧が表の中に記載されるべきである。もちろん強度を増すには多重暗号化が必要であり、そのときは適用するアルゴリズムの順番が大切になる。アルゴリズムは適用される順番で記載されなくてはならない。
もちろん、専門家の意見が対立する場合もある。それぞれが自信を持って使うアルゴリズムは異なる場合がある。したがって、表は私が主張する拡張アドレス帳の形、さらにその多重暗号化の場合の形にするべきである。

４．について
鍵の長さの制限は、必要とする暗号の強度を与えてくれない。アルゴリズムの制限は解読技術の進歩に対応できないものを提供することになる。国内で暗号の制限をすると、新しい技術を開発しても使えないので、その国で暗号の研究をしても経済的な利益がでない。暗号研究者はその国を去り、自由に研究し自由に販売が出

来る国に移るであろう、当然暗号研究が盛んな国では解読技術の研究も進む。制限した国の古い暗号は簡単に解読されてしまうようになる。暗号を制限する国は国際的な情報戦争の場面で敗者の道をたどることになる。犯罪者に対しては、法律をにそってパソコンを押収すればよい。よい政治を行えば国民は全員が政府に協力し、その国の大切な情報をみんなで守り抜くであろう。この協力が同時に、アルゴリズム開発者の経済的な利益になるような仕組みが必要である。

５．について

この表はどこに置かれ誰が管理するのか？各ユーザーのパソコンの中に置く場合は、国の ID の管理がおろそかになる。新しい国が出来たことを気にする人はあまりいない。新しい暗号が出来たときその ID を正確に記入することが、全ての人に可能だろうか誤った入力は暗号解読を妨げる。暗号通信が出来なくなる。

国のサーバーを決めてそこで一括管理する案では、個人のパソコンから国のサーバーまでの途中で盗聴されると困る。また、個人が勝手にこの表にない方法で暗号化してそれを添付ファイルで送ろうとしたら、その添付ファイルが暗号化されたものか否かを確かめるのか簡単ではない。送信者が、３２ビット整数の１６進数で表した乱数列のバイナリデータです。といい張ったときには、解読できない限りこの主張を覆す方法はない。したがって、国ごとの制限を付けても無駄になる。

困ったことはまだある、暗号化ソフトは何本売れるだろうか？国ごとに１つのサーバーを置いてそこで暗号化を処理すると、暗号化ソフトは国の数しか売れない。国が高い値段で買ってくれるかどうか分からない。開発したのに自分のソフトの売り込みに失敗した人は、使われている暗号ソフトの解読方法を開発し無料で配るでしょう。理由は自分の暗号ソフトを採用してもらいたいから。暗号開発者の経済的な保証をの問題が生じる。

また、ファイル名だけではその内容は分からない。テキストファイルかどうかは開いてみなくては分からない。全ての添付ファイルを開いて確認しないと、違法な暗号化が行われていないかどうかは分からない。一見しただけでは、平文に見える暗号文もある。この場合はさらに困ってしまう。

これらの観点から、この方法は現実性のない、効果のない方法である。といえる。私の方法を利用すべきである。

For inter-country communications, the key recovery system can determine the receiver's country ID from his public key certificate or comparable system configuration information. Using the sender's origin country ID and the receiver's destination country ID, and the algorithm ID of the intended encryption algorithm to be used by the sender and the recovery, the key recovery system will determine the following: (1)the maximum key length that the sender and the recover can us, (2)the allowed R values, which can be different for the sender and the receiver, and (3)the required key recovery agent IDs needed by the key inversion function. The key recovery system then passes this information to key inversion function. The key length is the smaller of the two key length values. For example, for countries X and Y the key values for DES are 64 and 128 bits, in which case 64 if the value used.

国際間の情報通信では、鍵復元機構では、受信者の国識別情報を彼の公開鍵の証明書または比較できる仕組み構成情報からから決定する。彼の鍵証明書、または同等のシステムの配列情報から受信者の国の識別記号を決定する。送信者の元の国の識別記号を使って、受信者の宛先国の識別記号と、送信者と受信者が使おうとしている暗号化アルゴリズムのアルゴリズム ID を決定します。鍵復元システムは次のことを決定してゆきます。（１）送信者と受信者が使用できる最大の鍵の長さ。（２）可能なＲの値、送信者と受信者で異なる値。（３）鍵復元窓口の識別番号鍵を反転する関数に鍵復元機構はこの情報を渡す。例えば、Ｘ国とＹ国とに対して、DES の鍵の値は６４または１２８ビットです。この場合では６４ビットとなる。

＜反論＞

この話の中では、国際的な協定でアルゴリズム ID が決まっていることが必要となる。X 国で新しいアルゴリズムが発見され、その解読方法を X 国の人だけが知っているとき、X 国はこのアルゴリズムを使いたいと思うであろう。この暗号が解読できない Y 国は、このアルゴリズムに ID をつけて使うことに反対するであろう。X 国は Y 国に解読方法を教えるのだろうか？国際的な協力によって ID を決めるためには解読方法を教えなくてはならない。当然全ての国に解読方法を教えなくてはならない。理由は国際的に通用する ID を決めるため。問題は情報公開をする国があることです。解読方法はインターネットを通じて全世界の人々に広まります。これはもはや暗号ではない。暗号化アルゴリズムの ID は意味のないものである。

国際間の通信、特にメールは SMTP サーバーを利用している。現在のメールサーバー（SMTP）には、こ

こで説明された機能はない。これを実現するには世界中のメールサーバーを全て変更しなくてはならない。その費用の負担は誰がするのか？メールを利用するものが自らの通信内容を公開するためにその費用を負担するのだろうか？

さらに、新しい国がどんどん成立している。それらの各国の法律の変更が正確に新しいメールサーバーに反映できるとは思えない。その意味では実現性のない案である。

私の方式は、添付ファイルとして送ればどんな内容のものも送信可能である、という現在の SMTP サーバーの形式をそのまま利用している。世界のサーバーの変更は全く必要ない。

一般に TCP/IP を使ったインターネットの通信ではパケットのヘッダー部分に宛先情報を付けて送信し、途中でその値を解析し、伝達する時のルートを決定するのに表が使われます。一般のアドレス帳も表形式になっています。その表が何を実現するためにどのような特徴を持っているのかを考えて区別する必要があります。

文献＜Ｄ２＞

column 10,line 33 - column 11, line 27

The UCDB 226 (or 226a) is conceptually illustrated in FIG.4. The heading in each column are not actually stored in the database 226, but are provided in GIG.4 for illustrative purposes. For each correspondent address 402, the UCDB 226 may have one more of following entries: own channel 404, correspondent channel 406, own key 408, and correspondent key 410. A correspondent is simply another e-mail user that the user desires to send to or receive mail from. The correspondent address 402 is the standard e-mail address of the correspondent. The own channel 404 is the channel ID 110 used for the correspondent to send e-mail to the user. The correspondent channel 406is the channel ID 110 that must be used by the user to send e-mail to the correspondent. The own key 408 is a key assigned by the user that is necessary to allow certain functions to be performed (as will be discussed below). The correspondent key 410 is a key assigned by the correspondent, which is also necessary to allow certain functions to be performed (as will also be discussed below). All of this information is placed in the UCDB 226 by the PCA 214 as it interacts with the mail server 202 and the user machine 206.

So, for example, as illustrated in FIG.4, for the correspondent address 402 “frobboz@geewhiz.com”, the following information is stored in the UCDB 226: own channel 404=1QXR7112PH; correspondent channel 406=1DDYWA7H71; own key 408=X9GWAAH4T; and correspondent key 410=T7AA18MMN, For the correspondent address 402 “jrandom@j.r.isp.ner”, all of the entries are similarly filled except for the correspondent key entry 410. This is because the correspondent key information has not yet been received from the correspondent, possible because it is not needed. Nevertheless, the UCDB 226 can store this correspondent key information when (and if) it is received.

The first two own channels 404 listed in FIG.4, “1QXR7112PH” and “122PG3LWAW”, begin with the number “1”. This number is the channel class indicator 116 discussed above and preferably indicated above. The next two own channels 404 listed in FIG.4, “2AA47WX3BQ”, and “2BZZ86A9FC”, begin with the number “2”. This channel class indicator 116 identifies these channels as public channel as described above. Therefore, these two channel do not require entried for correspondent address 402, correspondent channel 406, own key 408, or correspondent key 410, because any one of a number of correspondents may send e-mail on these public channels (using the public channel ID). The correspondent channel 406 associated with the correspondent address 402 “infohooha@sri.com” shown in FIG.4 is “0XTRR9Y680”. Because this channel class indicator 116 begins with the number “0”, this indicates that this channel is a send-only channel as described above. Therefore, there is no need for an entry for correspondent channel 404 or correspondent key 410 because no return messages will be accepted on this channel from any correspondent. In one embodiment, there is an own key entry 408 (in this case “BZTGRTKI3”) because the system automatically generates an own key even time a new correspondent is entered into the UCDB. In the case of a sender-only channel, this key would not be used.

図４は、UCDB（User Channel Detabase） 226 の概念を表現したものです。
各列の先頭は、実際にデータベース２２６に含まれるわけではないが、図４では、説明のためにあえて記載しています。UCDB２２６では、４０２のそれぞれのアドレスに対応して、それ自身のチャンネル４０４，それに対応するチャンネル４０６，それ自身のキー４０８と，それに対応する鍵４１０のように、引き続いているもののうちの一つまたはそれ以上が存在する。

対応は単純です、他の電子メールのユーザーへ送信またはそのユーザーからのメールを受信しようと思ったとき。対応するアドレス４０２は通信相手の標準的な電子メールアドレスです。

独特なチャンネル４０４は、チャンネル識別子１１０と同じですが、これは、相手がそのユーザーに電子メールを送るのに使われる。

相手側のチャンネル４０６は、チャンネル識別子１１０と同じですが、これはユーザーが相手に電子メールを送るのに使われる。

自分の鍵４０８は、ユーザーによって指定されたものであり、後で述べるが、これは、ある関数が動きを完了するために必要となる。（詳しくは後で述べる）

相手の鍵４１０は、相手から指定された鍵であり、ある関数の機能を果たすのに必要となる。（詳しくは後で述べる）

これらの全ての情報は、UCDB226 におかれ、PCA214 によって、メールサーバー２０２，ユーザマシーン２０６と情報交換する。

たとえば、図４に有るように、相手方アドレス４０２が、”frobboz@geewhiz.com”とする。UCDB226 に保存された情報、自分のチャンネル４０４＝1QXR7112PH;相手のチャンネル 406=1DDYWA7H7I;自分のキー４０８＝X9GWAAH4T;相手のキー４１０＝T7AA18MMN。相手方アドレス４０２が、“jrandom@j.r.isp.net”

の項目は相手方キー４１０を除けば、同様に書き込まれている。これは、相手方キーをまだ、相手から受け取っていないことを意味している。あるいは、必要がないことを意味している。UCDB226 は相手のキー情報を受け取り次第登録する。

図４にある自分のチャンネル４０４の項の最初の二つは”1QXR7112PH”,”122PG3LWAW”であり、どちらも、最初の数字は１で始まっている。この数値がチャンネルクラスを表している（１１６）。以前に議論したように、プライベートチャンネルである。

図４の４０４にある、次の２つの自分のチャンネルは、”2AA47WX3BQ”と”2BZZ86A9FC”であり、２の数字から始まっている。１１６にあるクラス識別子は、これらのチャンネルはパブリックチャンネルであることを表す。

これらの２つのチャンネルは、相手方アドレス４０２，相手方チャンネル４０６、自分のキー４０８，相手方キー４１０を持たない。それは、電子メールが、公開チャンネル ID を使って送られるからである。

相手方チャンネル４０６で、相手方アドレス４０２が”infohooha@sri.com”であるものは、図４では、“0XTRR9Y680”である。

このチャンネルクラス指示子１１６は、数“０”から始まっている、これは、このチャンネルが送信専用であることを意味する。したがって、相手方チャンネル４０４，相手方キー
４１０は必要ない。なぜなら、相手からの返信メールは一切受け付けないからである。

唯一記入されている項目４０８は、“ＢＺＴＧＲＴKI3”である。理由は、新しい通信相手が UCDB に加わったときに、システムが自動的に自分自身のキーを作成するからである。

＜反論＞

表の形は似ているが、この表の存在位置と有効性に疑問がある。現在スパムメールを防ぐのに、プロバイダーから与えられたアドレスの他に、ウエッブメールを使って、アドレスを複雑なものや、タイプのことなるものを用意し、使い分けるのは普通に行われていることである。

似ているのは表の列の数だけである。しかも、多重暗号化に関係のない基本形のみが列の数に関してのみ似ている。機能も使い方もまったく異なる。表形式だから新規制がないと言われると困るが、大切なのは表の項目に何を設定し、それをどの様に機能させるかという点である。

私の表は、ユーザーID、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵であり、これによって、順序ついに対応した暗号通信を実現する。さらに、多重暗号化も可能にし、意味ある暗号通信を実現する。もちろん必要なのは分離型暗号ソフトである。

単に迷惑メールを防ぐだけなら、無料のメールアドレスを沢山持って、しかも全て複雑なものにしておけばよい。しかしこれも、プロバイダーからアドレスが大量に流出し、犯罪的な形で販売されるとすぐにスパムメールが来るようになる。もちろん新しい無用メールアドレスをとり、友人に知らせればよい。

この方法の欠点は、個人または、会社で特殊な PCAHOST206 を所有または管理する必要があり、この PCAHOST206 はメールを受信したりアドレスを変換したりする特殊なサーバーである。メールアドレスの変更可能なサーバーを運用するのは個人では経済的に無理がある。ある会社で利用する場合、固定アドレスを取得し、サーバーを運営し、各ユーザーに幾種類かのアドレスを与えることも可能であろう。その結果、その会社の誰が暗号通信をしたかは隠せてもその会社から暗号通信が行われたことは隠せない。盗聴のターゲットが個人に絞られるのかそれとも会社に絞られるのかの違いしかない。盗聴者は会社の人数分の通信を解読すれば済むので、大した手間ではない。必要なら、社員の人数分だけ解読用のコンピュータを用意すればよい。この方法は暗号強度を特に高める要素を持たないので、情報を防衛する方法としては意味をなさない。

暗号化されないデータをインターネットに流すことは、裸で歩くのと同じである。洋服に当たるのが暗号化である。みんなが洋服を着るのと同じように、暗号化の着物を着せてデータをインターネットにながすべきであり、全員が暗号化するようにすれはよいのであり、誰のデータが暗号化されたかを知っても意味のない世界がすぐに来る。全員が暗号化されたデータを流すようになる。

普通の人は、数個のメールアドレスを使うことが多い。このとき、スパムメール対策でもっとも有効なのは、アドレス帳に登録されていない人からのメールは全て削除することである。メールサーバーには来ていても、自分のコンピューターにダウンロードする過程で、発信元のアドレスが解析された時点でアドレス帳とデータ比較をして、削除するか最後までダウンロードするかを決めるようにすればよい。アドレス帳に登録されていないが本当にメールを送りたい人は、前もって電話か手紙で、自動的に削除する機能を OFF にしておいてもらえばよい。

Referring now to FIG.9 and FIGS. 10A, 10B, 10C, 10D and 10E, the process for switching a channel when a user and a correspondent both have PCAs will now be described. In FIG.9, the letter “A” refers to a user who wishes to change a channel that a correspondent, referred to as “B” is represented by “frobboz1DDYWA7H7I-@geewhiz.com”. Similary, in FIGS. 10A through 10E, user A is represented by “someone-1QXR7112PH-@domain.com”. In addition, FIG.2 should also be referred to wherein it will be assumed by way of example that A is using user machine 206 and B is using user machine 206a.

Once a user (A) has decided that he wants to switch the channel that a correspondent (B) user to communicate with the user (A), the system first, in step 901, creates and opens a new channel to be switched to. The new channel is placed in the channels file. Next the system determines in step 902 whether correspondent B’s key is located in user A’s UCDB 226. In particular, the ley would be located in user A’s UCDB 226. In paticulare, the key would be located in the “correspondent key” filed 410. If not, user A’s PCA 214 sends a message to correspondent B’s PCA 214a requesting correspondent B’s key in step 904. The message also includes user A’s key so that correspondent B can include it in the replay. This message is illustrated in FIG.10A wherein a “send your key” message 1001 is sent. User A’s key 1002 is included in this message.

Next, in step 906, correspondent B’s PCA 214A sends user A’s PCA 214 correspondent B’s key and includes user A’s key in the message. User A’s key has been copied from user A’s previous message (FIG.10A). This allows user A to know that correspondent B’s key are taken from B’s UCDB 226a. If B’s UCDB 226a has no entry for user A, the message in step 9006 is not sent. FIG. 10B illustrates the message in step 906. Correspondent B is sending a “my key is” message 1004 to user A so that user A acquirres correspondent B’s key, in this case “T7AA18MMN”. In the message, user A’s key 1006 is included for authentication. User A’s PCA 214 records correspondent B’s key in user A’s UCDB 226 in step 908.

The system next produced to step 910, which is also where the system would proceed if in step 902 it was determined that correspondent B’s key was already located in user A’s UCDB 226. In step 910, user A’s PCA 214 sends a message to correspondent B’s PVA 214a to switch t a new channel. User A’s PCA 214 sends a copy of correspondent B’s key as well as A’s own key in this message. Such a message is illustrated in FGI. 10C, . In FIG. 10C the message to change channel to a new channel 1008, in this case “1L3RG359T”, is forwarded to correspondent B. Included with this message is a copy of correspondent B’s key 1010 so that user B knows correspondent A’s message is legitimate. A’s key is also included 1006.

Next, in step 912, correspondent B’s PCA 214a changes correspondent B’s UCDB 226a to reflect the new channel to be used to correspond with user A. Also, in step 912, correspondent B’s PCA 214a sends an acknowledgment on the old channel to channel to user A that the channel has been changed. The acknowledgement message includes a copy of user B’s key for authentication as well as a copy of A’s key. This type of message is illustrated in FIG. 10D. In FIG. 10D, an acknowledgement message 1011 is sent along with user B’s key 1010 and A’s key 1006 as well as the new channel 1012.

Next, in step 914, user A’s PCA 214 closes the old channel for correspondent B and sets the new channel as B’s channel in user A’s UCDB 226. Following this, an exemplary message that could be sent from user A to correspondent B is illustrated in FIG. 10E. It should be noted that the new channel identifier 1016 is placed in the “To” field for sending to the user (A) who has switched channels.

This protocol illustrated in FIG. 9 can be used to detect eavesdroppers. For example, if user A’s PCA 214 receives a second “my key is” message (for example like that shown in FIG.10B) indicating correspondent B’s key, as in step 906, which message includes user A’s valid key, but a different key is indicated for correspondent B, then user A knows that an eavesdropper has sent one of two messages. Thus, an eavesdropper has been detected and user A may alert correspondent B of this situation. In addition, if user A’s PCA 214 receivers a “my key is” message (for example like that shown in FIG. 10B) (as in step 906) providing correspondent B’s key that is not expected (i.e., step 904 has not occurred), then an eavesdropper detection can also be entered. Finally, if user A’s PCA 214 receives an acknowledgement that a change in channels has occurred (for example like that shown in FIG. 10D) with a valid key from correspondent B, but the change in channel is unexpected, then an eavesdropper detection condition is also entered. When an eavesdropper detection condition is entered, user A may notify correspondent B and appropriate action may be taken. This notification preferably should include correspondent B’s key so that correspondent B will know the message is legitimate. This notification could also tale place outside of the e-mail environment, for example, by telephone.

図９，１０A,10B,10C,10D,10E は、送信者と受信者の両方が PCA である場合、チャンネルを切り替えるプロセスを説明しています。

図９の文字”A”は、通信相手である“Ｂ”が“A”に対してメールを送るときのチャンネルを変えようとしている人を表すとします。

図 10A から、10E において、”B”は、”FROBBOZ-1ddywa7h7i-@GEEWHIZ.COM”で表されます。

図 10A から 10E では、”A”は、”somoone-1QXR7112PH-@domain.com”で表されます。
さらに、図２では、Ａは、２０６の機械をつかい、Ｂは２０６a の機械を使うとします。
（A）は、（B）が（A）との情報交換用使っているチャンネルを変更しようと決めたとします。
システムは、ステップ９０１で、変更用の新しいチャンネルを生成します。新しいチャンネルは、チャンネルファイルに置かれます。
次にシステムは、ステップ９０２で、通信相手の（B）の鍵が、Ａの UCDEB226 の中に有るかどうか確認します。
特に、鍵はフィールド４１０すなわち相手方鍵の項目にあるべきです。
そうでなければ、Ａの PCA214 は、相手方Ｂの PCA214a にメッセージを送ります。ステップ９０４において、相手方Ｂの鍵を要求します。
そのメッセージはＡの鍵も含んでいます。これによってＢは返信の中にそれを含めることが出来ます。この様子は図１０Ａで説明されています。１００１はそちらの鍵を送って下さいというメッセージであり、ユーザーＡの鍵１００２がこのメッセージに含まれています。
次のステップ９０６では、通信相手であるＢの PCA214ＡはＡの PCA214 に対して、Ｂの鍵と、Ａの鍵を含んだメッセージを送ります。
ユーザーＡの鍵はＡの前のメッセージからコピーされます。図１０Ａ。
これによって、Ａは相手方Ｂからのメッセージが正式なものであると知ります。
ステップ９０６で、Ａのチャンネル化されたアドレスと、Ｂの鍵は、Ｂの UCDB２２６a からとられます。
もし、Ｂの UCDB226a にＡの項目が無いときは、ステップ９０６のメッセージは送られません。
図１０Ｂでは、ステップ９０６で、メッセージが送られる場合を示しています。
通信相手Ｂは、Ａへの“私の鍵は“のメッセージ１００４を送り、これによってＡは通信相手Ｂの鍵を入手します。ここでは”T7AA18MMN”です。
このメッセージの中に、ユーザーＡの鍵１００６が、認証のために含まれています。
ユーザーＡの PCA214 は､ステップ９０８において､通信相手Ｂの鍵をＡの UCDB２２６の中に記入します。
システムは次にステップ９１０に進みます。もちろんステップ９０２で、Ａの UCDB226 の中に、すでに通信相手Ｂの鍵が含まれていた場合にもこの９１０に進むことが決められていました。
ステップ９１０において、ユーザーＡの PCA214 は通信相手Ｂの PCA214a にメッセージを送り、新しいチャンネルに変更するように要求します。
ユーサーＡの PCA214 はそのメッセージの中に、ユーザーＢの鍵のコピーをＡ自信の鍵のこーぴーと同様に含めて送ります。
このようなメッセージは、図１０Ｃで表現されています。
図１０Ｃでは、メッセージはチャンネルを新しいチャンネル１００８に変えます。この場合”1L3RG3592T”が通信相手Ｂに対して送られます。
このメッセージに含まれて、通信相手Ｂの鍵のコピー１０１０も含まれます。これにより、Ｂは通信相手Ａのメッセージが正当なものであると認識します。
Ａの鍵は１００６に含まれています。
つぎに、ステップ９１２では、通信相手Ｂの PCA214a はＢの UCDB226a を変更し、通信相手Ａに対応する新しいチャンネルを反映させるように変更する。
さらにまた、ステップ９１２では、通信相手Ｂの PCA214a はユーザーＡの古いチャンネルに対して、チャンネルがすでに変更されているとのお知らせを送信する。
お知らせのメッセージには、Ｂの鍵のコピーがＡの鍵のコピーと同様に、認証のために含まれている。
この形式のメッセージは図１０Ｄで表現されています。
図１０Ｄにおいて、受け取り通知のメッセージ１０１１がＢの鍵１０１０とＡの鍵１００６だけでなく新しいチャンネル１０１２と共に送られます。
次のステップ９１４では、Ａの PCA214 は通信相手Ｂに対する古いチャンネルを閉じます。そして、新しいチャンネルをＡの UCDB226 におけるＢのチャンネルとして設定します。
これに続き、ＡからＢに模範となるメッセージが送られます。この様子は図１０E に描かれています。
次の点は注意が必要です、新しいチャンネル識別子１０１６はチャンネルを変更したＡに対する“To”の項に置かれています。
図９に描かれた手順は盗聴を防ぐのに使われます。
例えば、ユーザーＡの PCA214 が、２番目の“私の鍵は”メッセージを受け取ったとき、(例は図１０Ｂに有ります）ステップ９０６で、指示されているＢの鍵、そのメッセージはＡの鍵を含んでいる。しかし、Ｂに対する異なる鍵を含んでいる。Ａは盗聴がこの二つのメッセージに対してなされたことを知る。
これによって盗聴は防がれる。ユーザーＡはＢに対してこの状況に関する警鐘をならす。
さらに、Ａの PCA214 が、ステップ９０６での図１０Ｂにあるように、“私の鍵は”メッセージが、通信相手

Ｂの鍵で、予測していないものを受け取ったとき、盗聴者の発覚が記録される。
　最後にユーザーＡの PCA214 が、図１０Ｄに有るような、通信相手Ｂの価値有る鍵と一緒にチャンネルが変更されたとのメッセージを受け取ったにもかかわらず、チャンネルの変更が予期されないものだった場合には、盗聴者の発覚が記録される。
　盗聴者発覚が記録されたとき、ユーザーＡは、通信相手Ｂに連絡をする。適当な対応がとられる。
　この通知には、Ｂの鍵が含まれるべきである。これによってＢはこの通知が本物だと分かる。これは、電話などの電子メール以外の方法で行われるべきです。

＜反論＞
　ネットワークでの傍聴が可能な人に対して、チャンネルの切り替えは意味を持たない。傍聴する人や組織はあらゆる通信チャンネルを傍受できると考えるべきである。
　さらに、現状では、普通の電子メール使用者はプロバイダーから与えられたアドレス幾つか持っている。これらはプロバイダーから与えられたものであり、自由に変更することは出来ない。ましてや、これを通信の途中で切り替えることは出来ない。
　しかも、電子メールの良さは、送信者は自分の都合の良い時間にパソコンのスイッチを入れメールを作成し送信後パソコンの電源を切る。送信されたメールは SMTP サーバーを経由してプロバイダーのメールボックスに入る。受信者は自分の都合の良い時間にパソコンの電源を入れ、プロバイダーのサーバーのメールボックスにあるメールを取り出す。
　普通パソコンは電源が切れている。メールはプロバイダーの POP3 サーバーにある。
直接ユーザー（Ａ）とユーザー（Ｂ）のパソコンが通信を行うことはない。
そこで、問題となるのは、
　（Ａ）の通信相手Ｂの UCDB の存在する場所はどこか？

１．（Ｂ）のパソコンの中なら、電源が切れているときは、電話して電源を入れてもらわなくてはならない。電子メールの本質、相手が自分の都合の良いときに自分のパソコン電源を入れてメールを取り出す。相手からの夜中の電話に起こされることもない。国際間のメールが一般手にな現在、不便きわまりない。また、電源を入れたままのパソコンはハッカーの攻撃対象になる。不安が増すだけである。
　電話して電源を入れてもらうなら、その電話で話をしてしまえばよい。もし電話の盗聴が心配なら、その盗聴を防げばよい。

２．プロバイダーのサーバーの中か？、いまの SMTP サーバーや、POP3 サーバーはこの機能を持っていない。新しいサーバーがこの機能を持ったとして、世界中のプロバイダーにこの各個人のデータベースを保存すると、個人の友好関係が全て他の人に把握されてしまう。これは、個人のプライバシーの崩壊につながる。

３．メールサーバーを立てた会社ごとにあるのか。会社の誰からのメールかを隠しても意味がない。個人の顧客は、この変なサーバーに対応していないので、作っても効果はない。ハッキングの目標を作り出すだけである。

　どちらにしても、魅力はない。

column 17,line 62 - column 19, line 14

Referring to FIG. 12, Packet Telephony may be used over a network 1202. There are at least two exemplary possibilities for using PCA's in combination with such a network 1202 includes a Packet Telephony. The first embodiment may be used with a personal computer ("PC") 1204. The PC 1204 includes a Packet Telephony server 1206 as 1206 as is known in the art, but modified as described below with reference to FIG. 13. The Packet Telephony server 1206 may preferably communicate with a voice input/output interface 1208. This voice input/output interface 1208 may include a speaker and microphone as is known in the art. The voice input/output interface 1208 is accessible to user 1210 for "talking to" and "listening to" toher users of the network 1202. The user would also have access to a web browser 1212, which may be used to administer the PCA 1214 as described with respect to the previously-descrined embodiments. The PCA 1214 also communicates with and modifies and updates a chanels file 1216. The PCA 1214 also communicates with the Server 1206. The channels file 1216 is read by the Packet Telephony

server 1206.

A telephone call may preferably be initiated useing the web broser 1212 by proving a (place call" button on an administrative interface (similar to that shown in FIG. 11). A user would select a correspondent entry from a correspondent address map display and then the "place call" button.

Alternatively, a user could use a stnderd telephone 1218 to dial up a PCA 1214a with is located remotely on a machine or computer 1217 at a service provider's facility. Using standard dual tone multifrequency ("DTMF") touch tones the user can place telephone call with the telephone 1218. To administer the PCA 1214a so that if updates or changes a channels file 1216a or a UCDB, the user may access the PCA remotely through a standard web browser 1212a as described for the embosiment show by 1204, except that a "place call"button would not be needed. This web browser 1212a may be located on a user's PC 1219. Again, as with the embodiment shown by 1204, the Packet Telephony server would be modified as described below.

The Packet Telephony servers 1206, 1206a may communicate with the network 1202 as is known in the art. In both embodiments 1204, 1217, the Packet Telephony server 1206, 1206a would refer to the channels files 1216, 1216a to determine whether or not an incoming call is to be accepted. In addition, although not shown in FIG. 12, the PCA's 1214, 1214a would each have a UCDB which would be the same as for the other embodiments described earlier, For outgoing calls, the UCDB may be referenced to add the appropriate channel ID (if any) to the intended recipient's address.

In FIG. 12, a "third party" 1220 is shown for use in explaining how telephone call may be sent to and received from that third party 1220.

Referring now to FIG. 13 the process for receiving a packet telephone call, for example from third party 1220 will now be described. In this examole, it is assumed that the cakk is directed to user 1210, but an indication of how the call would proceed to the telephone 1218 is also indicated parenthetically. In step 1301, the caller (third party 1220) transmits a call set-up request in cludeing the user's address to the recipients/user's telephony server 1206 (or 1206a). This will include the user's channel ID if the user uses channelized address for incoming calls.

In step 1302, the user's telephony server 1206 (or 1206a) checks the address sent by the third party 1220 to determine if the user it designates is an authorized user 1210 (or 1218) for that system. If in step 1302 the server determines that the address does not designate an authorized user 1210 (or 1218), the system refuses the call setup request in step 1303. If it is determined in step 1302 that the user is an authorized user, then the process proceed to step 1304.

In step 1304, the server 1206 (or 1206a) checks the channels file 1216 (or 1216a) and determines whether or not the channel ID provided by the third party 1220 is open. If the channel ID is not open, in step 1306, the system will refuse the call set-up request from the third party. If, in step 1304 it is determined that the channel provided by the third party is valid and found as open channel in the channels tile 1216 (or 1216a) the server1206 (or 1206a) will, in step 1308 set-up the call in a conventional manner. This will include in the embodiment shown by 1204 sending the information to the voice input/output interface 1208 so that the call may be listened to and responded to by the user 1210. In the embodiment shown by 1217 through the PCA 1214a so that the user can pick up the telephone and communicate with the third party 1220.

図１２での、ネットワーク１２０２上で使われるパケット電話にかんしては、少なくとも、２つの可能性が例示してある。PCA とネットワーク１２０２を組み合わせたパケット電話の例が示してある。

最初の例は、パソコン PC１２０４を使って実現される。PC1204 はパケット電話サーバー１２０６を含み、すでに技術的には知られているものに図１３にあるような変更を加える。

パケット電話サーバー１２０６は、音声の入出力インターフェイス１２０８と好んでコミュニケートする。

この音声入出力インターフェイス１２０８は、工学的なマイクロフォンとスピーカーを含んでいます。

音声入出力インターフェイス１２０８はユーザーがネットワーク上のユーザー１２０２と話しかけたり、聞いたりするのに利用できる。

ユーザーは、ウエブブラウザ１２１２も利用できる。これは、PCA1214 を管理するのに使われる。前もって記述されたものを具体化する。

PCA１２１４は、チャンネルファイル１２１６を更新したり変更したり、とコミュニケーションを取る。

チャンネルファイル１２１６は、パケット電話サーバー１２０６によって読み込まれる。

電話呼び出しは、ウエッブブラウザ１２１２を使って開始されるのが望ましい。管理のためのインターフェイスに用意してある、場所を呼び出すボタンを押して。(図１１と同様)

ユーザーは相手のアドレスから、相手の項目を選び、場所呼び出しボタンを押す。

その代わりに、ユーザーは普通の電話１２１８を使って、サービスプロバイダーの設備によって、遠方にある機械またはコンピュータの中にある、PCA1214a に電話をかけることも出来る。標準的な両方向多重周波数(DTMF)のプッシュボタンの電話をつかって、電話機１２１８に電話をかけることができる。

PCA1214a を管理し、チャンネルファイル１２１６a または UCDB を更新または変更するのに、ユーザーは標準的なウエッブブラウザ１２１２a を使って遠い場所から、PCA にアクセスする。１２０４として具体的に示されているように、違いは場所呼び出しのボタンが押されないことを除けば同じである。

このウエッブブラウザ１２１２a は、ユーザーのコンピュータ PC1219 上に有るかも知れない。

１２０４で示してあるように、パケット電話サーバーは製品化に当たっていかに記述するように少し変更します。

パケット電話サーバー１２０６，１２０６a は、製品となっているものは、ネットワーク１２０２とコミュニケーションをとる。

１２０４，１２１７の両方で具体化されているものは、パケット電話サーバー１２０６，１２０６a は、チャンネルファイル１２１６，１２１６a を使って、かかってきた電話を受け取るべきか拒否すべきかを決める。

図１２にはないが、PCA の 1214,1214a はどちらも UCDB をもっていて、以前記述した場合と同様になります。

出発の呼び出しに対して、UCDB は新しいチャンネル ID を追加することに関係する。意図された受取手のアドレスに対して。

図１２は第３者１２２０との、電話呼び出しが発せられたり受け取られたしするのかを示しています。

図１３に関しては、第３者１２２０からのパケット電話の受信の様子を示しています。この例では、電話は１２１０に対してかけられるとしています。どのようにして、呼び出しが電話機１２１８まで達するかについても説明してあります。

ステップ１３０１では、電話をかける人（第３者１２２０）は、ユーザーのアドレスと、受信者の電話サーバー１２０６（または１２０６a）に対して、呼び出し確率の要求をします。

ユーザーが入ってくる呼び出しに対しては、チャンネル化されたアドレスを使うときには、ここに、ユーザーのチャンネル ID が含まれます。

ステップ１３０２では、ユーザーの電話サーバー１２０６（１２０６a）が、第３者から送られてきたアドレスをチェックします。このシステムに対して、権限のあるユーザー１２１０（または１２１８）なのかどうかの印となる。

もし、ステップ１３０２で、アドレスが権限のあるユーザー１２１０（または１２１８）を示さないとサーバーが決定したら、このシステムはステップ１３０３のように、呼び出し確立を中止します。

ステップ１３０２で決まったら、ユーザーは権限のあるユーザーです。過程はステップ１３０４に進みます。

ステップ１３０４では、サーバー１２０６（１２０６a）は、チャンネルファイル１２１６（１２１６a）を参照して、第３者１２２０によって与えられたチャンネル ID を開くべきか否かを決めます。

もし、チャンネル ID が開かれていなかったら、ステップ１３０６で、システムは、第３者からの呼び出し確立の要求を断る。

もし、ステップ１３０４で、第３者によって与えられたチャンネルが正式のものであり、チャンネルファイル１２１６（１２１６a）とサーバー１２０６（１２０６a）の中に開かれたチャンネルが見つかれば、ステップ１３０８で、形式に則って呼び出しがなされる。

これは、１２０４で示された情報が音声の入出力装置におくられ、ユーザー１２１０呼び出しが聞こえ応答することを含む。

１２１７で示す内容は、PCA1214a を通じてつながっているユーザーの電話１２１８が鳴り、ユーザーは電話機を取り上げで、第３者１２２０と話をする。

＜反論＞

チャンネルの切り替えに何の意味があるのだろうか？電話してくる相手もただ一つの電話番号しか持っていない。普通の人は、電話会社から与えられたただ一つの電話番号しか持っていない。使用する回線も変更できない。この条件の中で実現できるものを考えるべきである。

公衆電話網の中のユーザー権限で出来ることをはっきりさせると暗号化の形式がいかなるものであるべきかとうい課題の答えが見えてくる。

出来ることは、デジタル化された音声の代わりに、デジタル化された音声を同じビット数のデータに暗号化して、送信することである。利用者が必要とするだけの強度を与え、制作者の利益になるものを提案する必要がある。

利用するのは、分離型暗号ソフトと拡張アドレス帳である。これによって、パケットのなかにある、音声信号部分を暗号化して送れる。多重暗号化はユーザーが必要とするだけの、強度を与えてくれる。電話は全て受け入れるが、復号化が必要か否かが拡張電話番号表を使って決定される。もちろん復号化のアルゴリズム、復号化鍵も同時にきまる。

しかも、使用される分離型暗号ソフトは、暗号制作者の経済的利益になり、経済面から開発に貢献できる。

文献＜Ｄ３＞（sun microsystems inc)

page 4, line 42 - page 5, line 26

Figure 6 illustrates the method of the invention, and commences with the generation of data packets at the sending host A. The user at host A enters conventional commands for transmitting a file or the like from host A to host B, and the host computer A carries out the standard procedures for breaking the file down into data packets as in Figure 7, each including both the data 410 and a header 400. In the case of transmissions over the Internet, this will be the IP header. Though the current discussion will be detected in large part to IP-specific implementations, it should be understood that any network protocol may be used in conjunction with the present invention.

At box 200, the user at A (see Figures 3 and 6) enters the conventional command for sending the file, email, or the like to a recipient, and host A generates data packets for sending over the Internet in the normal fashion. Each data packet initially has a structure like that of data packet 400 shown in Figure 7, including a data field 410 and a header field 420. The header 420 includes the destination address, in this example the IP address of host B.

The data packets are transmitted by host A at box 210, again in conventional fashion. However, at box 220, each packets intercepted by the tunnelling bridge TB 1 (see Figures 3 and 4), when any of modes 2,2A, 3 or 3A is used (see discussion below). When mode 1 (described below) is used, step 220 and 280 are omitted, since this mode does not use tunneling bridges; instead, the actions taken by the tunneling bridges in modes 2-3A are all accomplished by the source and destination hosts themselves in mode 1. Thus, in the following discussion, where TB1 or TB2 is mentioned, it should be understood that in the case of mode 1, the same feature will be present in host A or host B, respectively.

Stored in the memory of TB1 (or host A, for mode 1) is a look-up table (not separately shown) of the addresses of hosts both on the local network N1 and on remote networks such as N2 and N3, and an indication for each network whether data packets from or to host should be encrypted. For instance, in this case the hosts table of TB1 indicates that any messages sent from host A to B should be encrypted. Thus, bridge TB1 (or host A) looks up hosts A and B in it's tables and determines that the data packets to be transmitted must first be encrypted, as indicated at boxes230 and 240 of Figure 6.

Alternatively the table could stored the network identifiers (e.g. broadcast addresses) of networks N1 and N2, indicating that anything sent from network N1 to network N2 is to be encrypted. In this case, the table need not list each host in each network, which makes the table smaller and easier to maintain.

If each host is listed, however, greater flexibility can be retained, since it may be that messages o or from particular hosts need or should not be encrypted. In an alternative embodiment, the look-up table list the networks N1 and N2 as networks to and from which packets should be encrypted, and also includes a hosts section of the table indicating exceptions to the normal encryption rules for these network. Thus, if networks N1 and N2 are listed in the look-up table, then packets travelling from N1 to N2 should normally be encrypted; however; if there is an "exceptions" suitable indicating that no packets from host A are to be encrypted, then the normal rule is superseded. The exceptions can, of course, go both ways where the normal rule is that the packets for a given network pair should/should not be encrypted, and the exception is that for this given host (source or recipient) or host pair, the packet should not /should nonetheless be encrypted in this embodiment, the small size and ease of maintenance of the networktables is by and large retained, while the flexibility of the hosts table is achieved.

If the data to be transmitted from host A to host B (or network N1 to network N2) should not be encrypted, then the method proceeds directly to step 270, and the packet in question is transmitted unencrypted to the destination, via the Internet (or other Intervening network).

In this example the packets are encrypted at box 250. This is carried out by the tunneling bridge TB1, according to whichever predetermined encryption scheme was selected, the primary requirement begin that of ensuring that TB2 is provided with the same encryption scheme so that it can decrypt the data packets. TB2 must also be provided in advance with the appropriate key or keys for decryption.

図６は、発明の方法を示している。送信するホストコンピュータＡのところで生成されるデータパケットから始まる。

ホストＡでのユーザーは、ファイル送信または、ホストＡからホストＢへの通信のために、規約通りの命令を入力する。そして、ホストコンピュータＡは図７にあるように、ファイルをデータパケットに分割する標準的な手順を行う。

それぞれは、データ部分４１０とヘッダー部分４００（４２０の誤りです。宇山）を含んでいる。インターネット経由の通信の場合、これはIPヘッダーである。

現在の議論は主に、IP に関して議論するが、どのようなネットワークプロトコルでも、この発明は使用で

きることを確認しておきます。
箱２００は、ホストＡ（図３，６）のユーザーは受信者にファイルや電子メール、などを送るために、規約通りの命令を入力する。ホストAは、インターネットに送り出すためのデータパケットを普通の形式で生成する。最初は、各データパケットは、図７でのデータパケット４００のような構造を持っている。データ部分４１０とヘッダー部分４２０を含んでいる。
ヘッダー４２０は宛先のアドレスを含んでいる。この例では、ホストＢの IP アドレスです。
データパケットは、箱２１０のところで、ホストAによって規約通りの形で送り出される。
箱２２０では、各パケットはトンネリング作成機 TB1 によって横取りされる。（図３，４）モード２，２Ａ、３または３Ａのどれかが使われる。（以下詳述する）
最初に、モード１が使われる場合を考えます。ステップ２２０と２８０は省略されます。なぜならこのモードはトンネル橋を使わないから。にもかかわらず、モード２－３Ａのトンネル橋における動きは、モード１での発信元ホストと送信先ホストの動きとして実現されます。よって、以下の議論では、どこに置かれた TB1 と TB2 について述べたとしても、モデル１の場合を含んでいると理解されるべきです。
ホストAとホストＢは同じ構造を持ちます。
TB1（モード１では、ホストA）は、メモリーにある参照テーブル（分離して表示してあるわけではない）で、ホストのローカルネットワーク N1 におけるアドレスや、遠くのネットワーク N2,N3 におけるホストのアドレスがあり、それぞれのネットワークについてそのホストからのデータパケットが暗号化されるべきか否かが示されている。
例えば、ホストのテーブル TB1 において、AからＢへ送られるどのようなメッセージも暗号化されるべきだと言うことが指示されている。しかるに、橋 TB1（またはホストA）はそのテーブルからホストAとホストＢを見つけ、送信されるデータパケットは暗号化されなくてはならないことを決める。図６の箱２３０，２４０参照。
また、そのテーブルは、ネットワーク N1 と N2 のネットワーク識別子（ブロードキャストアドレスのような）を含むことも出来る。これは、ネットワーク N1 から N2 へ送られる全てのものが暗号化されなくてはならないことを示している。この場合は、テーブルは各ネットワークにおける各ホストをリストアップする必要はない。こうすれば、テーブルが小さくなり、保守が楽になる。
もし、各ホストがリストアップされていれば、大きな柔軟性が維持される。というのは、特定のホストに宛てた、または特定のホストからのメッセージが暗号化されるべきかそうではないのかについて分かるから。
もう一方の具体化は、ルックアップテーブルに、ネットワーク N1 と N2 があり、ネットワークへのとネットワークからのパケットが普通は暗号化されるべきであるが、テーブルの中にホスト部分があり、これらのネットワークにおける標準の暗号化規則に関する例外規則が含まれている。
しかるに、ネットワーク N1 と N2 がルックアップテーブルにリストされていて、N1 から N2 へと送られるパケットは普通は暗号化されるべきであるとき、もし、例外規定があり、Aからのパケットは暗号化されないとなっているなら、正規の規則は適用除外される。
この例外は、両方の形で可能である。普通の規則をネットワークの組に対するパケットは暗号化（する／しない）とします。例外ルールを与えられたホストまたは、ホストの組に対してパケットを暗号化（しない／する）として具体化します。
例外規則を使うと、サイズが小さく、ネットワークテーブルのメンテナンスがしやすくなり、大きな記憶ホストテーブルの柔軟さが実現できる。
もし、AからＢへ（またはネットワーク N1 から N2 へ）送られるべきデータが暗号化されないとき、この方法ではステップ２７０に進む、問題のパケットは暗号化しないで相手方に送られる。インターネットまたは他のネットワークを通じて。
この例では、箱２５０でパケットは暗号化される。これは、トンネル橋 TB1 をとおる。前もって決められた方法で暗号化される。TB2 は暗号化されたデータパケットを復号化する必要性から存在する。TB2 には、前もって適当な鍵、復号化鍵を与えておく。

＜反論＞
参照テーブルを、サーバーに置くことは、順序対型の場合は不可能である。Aの通信相手は何人いるのだろうか、個人なら、１０００人分くらい置けば良いが、そのネットワークの中に世界中の人を相手に商売をしている会社が幾つか有ったなら、サーバーはこの会社のための参照テーブルをどの程度の大きさで確保するのであろうか？この大きさは世界中の人間の数と多重度と会社数の積で求まる。さらに問題がある、顧客データは大事な財産である。それをプロバイダーに預けることは、顧客情報の漏洩を引き起こす。会社の崩壊につな

がる。サーバーを借りたとしても顧客情報は会社内のコンピュータに置くしかない。この観点から見ても、この特許は方法としても、利用できるものではない。

その会社が自前のサーバーを建てると、使用できるのは与えられた IP アドレスであり、その IP からその会社が特定できる。トンネルの効果は半減する。

問題は参照テーブルの構造である。暗号の利用者は、情報の漏洩を防ぐためにどのような構造の参照テーブルを利用すべきなのか、が問題であり、私は拡張型アドレス帳としてその構造を明らかにしている。しかも、登録されるのは分離型暗号ソフトの各構成要素である。暗号選択の自由と鍵生成の自由さらに、多重暗号化に対応できる参照テーブル形が必要とされるのに、この特許では、それが示されていない。ネットワーク管理者の立場からの提案だからである。ここでの課題とすべきは、ネットワークに対する権限を持たない単なるユーザーに満足できる暗号化方法を提供することである。

一般のメールユーザーはホスト宛のメールを出さない。相手のメールアドレスに向けてメールを送信する。これが、相手のホストコンピュータに直接届くことはない。プロバイダーのメールボックスに入り、それを取り出すのである。ローカルネットワークのサーバーに関する権限は持っていない。

ましてや、ネットワーク間の暗号アルゴリズムの決定権は持っていない。管理者が直接の決定権を持っているとも思えない。理由は一つのローカルネットワークの管理者だけで新しい暗号アルゴリズムの採用を決めても、他のネットワークの管理者がその使用を認めない場合が十分考えられる。さらにそのローカルネットワークに属するユーザーがその暗号化アルゴリズムと鍵の長さについて十分だと思わないこともある

また、パケットが２１０から、２２０まで届く間に傍受されることも考えられる。完全に暗号化した後で公衆回線にデータパケットを送るべきであり、暗号化しないで送ったパケットを途中で暗号化するのは、平文と暗号文のセットを傍受者に与えることになる。これによって暗号文だけが傍受されたときの解読が簡単になる。

この観点では、２５０までが、ユーザーのパソコンで行われるべきであり、各ユーザーは全てネットワーク管理者の権限を持っていなくてはならない。このときは、各ユーザーが勝手に暗号化の参照テーブルを書き換えることになり、大きな混乱をもたらす。

どのアルゴリズムかは各個人が自由に判断するべきであり、ネットワーク全体での使用は解読されやすくなし、多重化や順序を各個人が変更することが出来なくなる。ネットワーク全体での利用は暗号の強度を下げる。暗号ソフトはあくまでも個人単位で、ばらばらに利用すべきである。

さらに、暗号ソフトの著作権と再配布に関して問題が生ずる。

“ＡからＢ”と“ＢからＡ”は区別されるべきである。

ネットワーク全体で管理するのはネットワーク管理者による盗聴と解読の可能性を生む。

管理者が何時悪人に変化するかは不明である。とても危険なやり方である。

復号化鍵は自分だけが持っているのがよい。ネットワークを通じて送ってはならない。

送ることは、暗号を解読しやすくするだけである。

普通の利用者はローカルネットワークの管理権を持たない。従って、ネットワークからネットワークへの暗号化は普通の人にとっては意味を持たない。

ネットワーク間で暗号化を統一すれば暗号が解きやすくなる。同一アルゴリズムで暗号化してある暗文が増えるからデータが増えて解読しやすくなる、例文が増えると統計的な処理がしやすくなる。ネットワーク内に平文を提供してくれる人がいればそれと暗文を比較して解読処理がやりやすくなる。

そもそも、２つのローカルネットワーク間の暗号化方式と鍵の長さを決める権利はどちらのローカルネットワークの管理者が持つのだろうか、自分だけが解読方法を知っていて相手の管理者が解読できない暗号化アルゴリズムと鍵の長さを選ぼうとするのが当然である。

特にローカルネットワークが一つの国であるときは国の利益が反映されるから。両方の管理者が合意できるのは、解きやすい暗号になってしまうであろう。暗号開発者と管理者の経済的癒着は当然生じる。３年後には廃れている暗号化が継続され、暗号としての意味を持っている新しいアルゴリズムは採用されない。

しかも、サーバーを変更する権限のないものにとっては、パケットのヘッダー部分の内容を勝手に変更することは出来ない。現在与えられている TCP/IP のプロトコルに従ったヘッダーを付けなければ通信は出来ない。

普通の、権限のないユーザーに、必要なだけの強さを持った暗号化の方法を提供することが必要である。これを実現するのが、分離型暗号ソフトと拡張アドレス帳である。これは、通信は２つの順序対であるとの考えから生まれる。

普通は、送信者と受信者は、プロバイダーから与えられたアドレスしか持っていない人間や、ローカルサーバーを立ち上げることがでいない人にとっては、“誰から誰への通信か”は隠すことが出来ない。

これを隠す必要はない。内容さえ分からなくすれば十分だと考える。

連絡と調整の手間な必要だが、分離型暗号ソフトの使用は、最新最強のアルゴリズムをすぐに利用で来る点において、その欠点を補ってくれる。

この手続きを忘れるような内容のものは、暗号化の必要はない。

page 6, line 15 - 24

In mode 2, a data structure 404 is used, and includes a new encapsulation header 432. It includes key encryption management information 440, which is appended to the original data packet 400, and both are encrypted, resulting in encrypted fields (410), (420) and (440) shown in Figure 9. A new IP header 470 including the broadcast addresses of the source and destination networks (not the addresses of the hosts, as in field 450 in Figure 8) is appended. In addition, a tunnelling bridge identifier field 460 is appended as part of the encapsulation header 432. Here, fields 410, 420 and 440 in this embodiment are all encrypted, while fields 460 and 470 are not.

The tunnelling bridge identifies the source tunnelling bridge, i.e. the tunnelling bridge at the network containing the host from which the packet was sent. The recipient tunnelling bridge contains a tunnelling bridge look-up table, indicating for each known tunnelling bridge any necessary information for decryption, most notably the decryption method and key.

モデル２では、データ構造４０４が用いられる。これは新しいカプセル化するヘッダー４３２を含む。それは、鍵の暗号化の取り扱いについての情報４４０を含む。これは最初のデータパケット４００に適応される。両方とも暗号化される。結果として、図９にある、暗号化された部分（４１０)、(４２０)、(４４０）となる。新しいIPヘッダー４７０には、送信元と宛先のネットワークについてのブロードキャストアドレスが入っている。(図８のフィールド４５０にあるような、ホストのアドレスではない)。さらに、トンネル橋識別子のフィールド４６０も入っている。これは、新しくカプセル化するヘッダー４３２の一部である。ここに有るフィールド４１０，４２０，４４０は、暗号化されているが、フィールド４６０，４７０は暗号化されていない。

トンネル橋識別子は、発信元のトンネリング橋を決定する。すなわち、ホストコンピュータがあるネットワークからパケットが送信されたことを示す。その、受信者のトンネル橋は、トンネル橋の参照テーブルを持っていて、各トンネル橋のと復号化に必要な情報を持っている。とくに、暗号化アルゴリズムと鍵の情報を持っている。

＜反論＞

自動的な復号化には、どこかに鍵と復号化プログラムが無くてはならない。問題は、そのアルゴリズムの採用と、鍵の長さの決定、異なるアルゴリズムでの多重暗号化による強化の権限を誰が持っているかである。

個人が持つなら、参照テーブルの存在場所が、個人のコンピュータの中に有るべきである。
個人ならば、どんなに大きくて複雑でもたかが知れている。計算がたいへんなら新しいPCを買えばよい。

つぎに、個人は、新しいヘッダーを付加えることが出来ない。通信経路を変えると言っても、実際には通信経路はそんなに多くはない。海底ケーブル、衛星通信、くらいであり、この通信を押さえることは経済力が有れば簡単です。自前の通信衛星と原子力潜水艦を持てば可能となる。さらにネットワーク管理者から情報を得れば、利用者が知らないうちに暗号が無意味になってしまう。

管理者が、暗号化についての情報を売り渡すのは別に不思議ではない。管理者が鍵の長さを短くするのは不思議ではない、サーバーにかかる復号化のための計算負荷を減らしたいと思うのは管理者として当然である。参照テーブルも小さくしたいであろう、時間が節約できるから。しかし多様な暗号化が有り、多重化が可能であってこそ解きにくくなるのだから参照テーブルは複雑で大きな方がよい。管理者の立場と暗号の強化を求める立場は相反する。

管理者が新たな暗号化アルゴリズムを採用することに積極的だとは思えない。参照テーブルの更新がサーバーに対する新たな負荷となるからです。

自分で暗号化について決定権のある個人が、自分の使う暗号についての情報を売り渡すことはないし、鍵の長さを短くすることもない。

（４４０）の部分は形が決まっていて、この部分が多重暗号化に対応できているか否かであるが、鍵が複数有ることは記述されていない。ましてやその順序の記述されていない。

さらに、XOR 方式のように鍵そのものが大きなファイルであるような場合は、この方式では対応できない。ヘッダー部分が５Mバイトあるようなパケットは送るのが困難である。従って XOR 方式には対応できない。ファイル名を送っても意味が無い。同じ名前で中身が異なるファイルが存在するのだから。

簡単で高速なXOR 方式が使えないのが大きな欠点である。

各論点についての反論

２．１　での不明確さのご指摘はその通りです。これは、コンピュータプログラムに係るものとして理解されなくてはなりません。
従って、請求項１については

1. A protected communication computer software applied to an information processing device such as a personal computer or the like having a memory and a communication function over the network, wherein the memory storing a reference table in which a relationship between a transmission destination data and an enciphering program, and a relationship between a transmission source data and a deciphering program being preset, and a plurality of kinds of enciphering programs and deciphering programs, comprising; means for automatically enciphering a transmission anticipated file according to an enciphering program upon an input of a prescribed enciphering instruction, the enciphering program being automatically specified by looking up the reference table using the designated transmission destination data as a key, and being read out from the memory of the information processing device; and means for automatically deciphering a received enciphered file according to a deciphering program upon an input of a prescribed deciphering instruction, the deciphering program being automatically specified by looking up the reference table using the transmission source data of a received enciphered file as a key, and being read out from the memory of the information processing device.

から、詳細を記述した部分を生かして、以下のように変更いたします。

１．A computer program stored on a computer-readable medium containing instructions which upon execution carries out the following method steps.
(1) This program has an extended address book. [Fig.1], [Fig.13]
(2) The extended address book must have those terms, name, address, encrypting key, encrypting software, decrypting key, and decrypting software. [Fig.1] , [Fig.13]
(3) The user of these software and key must use them under the 4 properties in [0071].
(4) In the case, when X send a e-mail to Y, the encrypting software and encrypting key in X's address book is given by Y. [0038], [0039]
(5) When X send a mail to Y, using table in FIg.1 or Fig.13, original data is encrypted. The encrypted data is attached to an empty mail. [0046], [Fig.2], [Fig.14]
It is clear that existing Internet can carry any type's file data. Base64 encoding make it possible. Reached data is base64 decoded. And encrypted data appears.
(6) In Y's extended address book, the term of X can decide the decrypting software and decrypting key. [Fig.1], [Fig.13]
(7) The mailer of protected communication system calls decrypting software, with decrypting key and encrypted data. [Fig.3], [0048], [Fig.14]
(8) The mailer of protected communication system gives us decrypted data. We can read and understand it.　[Fig.3]
These steps (1) to (8) shows basic movement of this system.
This character of Internet gives us powerful encryption. Next steps are needed.
(1) Encrypt data with enough strength. [0117]
Encrypted data is regarded as binary data.
Using base64 encode, this binary data changed to characters.
These characters are transposed to an addressed person.
Reached characters are base64 decoded.
(2) Given binary data is decrypted. Based on the extended address book. [0117]
This method gives us powerful encryption, without any change of Internet mail server, which existing now.
This software is the realization of ordered pair type's encrypting communication. For this realization next conditions are needed.
(1) Make an extended address book, as in Fig.1, Fig.13.
(2) Using separate type cipher software as in [0071].
(3) A receiver must decide and send encrypting software and encrypting key to sender. And the movement of system is described above.
If the concept of ordered pair is realized as ordered pair type encrypting communication system. Now

existing Internet giving us powerful method of encrypted communication. Without any change the structure of existing Internet.

２．２
請求項６を変更いたします。

現在の請求項６は
6. A telephone machine having a transmitter notification means such as a caller identification or the like, comprising; means for installing the enciphering program and deciphering program; memory means for storing enciphering reference data in which transmission destination data is pre-set in correspondence with an optional enciphering program; memory means for storing the deciphering reference data in which transmission source data is pre-set in correspondence with an optional deciphering program; means for enciphering and transmitting a transmission anticipated voice signal using a concerned enciphering program, said enciphering program to be used for the communication being identified based on the transmission destination data regarding the communication, and the enciphering reference data; and, means for deciphering and producing voice of the received voice signal in order using a concerned deciphering program, said deciphering program to be used being specified based on the data of a transmitter regarding the communication and the deciphering reference data.

ですが、これを、
6. A telephone machine is having a telephone number. This number is an identifier of the user. Instead of E-mail address, we can use the ordered pair of telephone numbers.
If a telephone machine have enough power for encrypting program. We can use ordered pair type's encrypting communication system on a telephone.
Cellular phone is not having such power now, but few years after it will have enough power.
IP phone on a computer can be used for realization of this type's telephone. It is easy to have a table of telephone number, enciphering program, enciphering key, deciphering program, and deciphering key, as in Fig.1, Fig.13.
Voice data is enciphered before transmission, using the table in Fig.1 or Fig.13, as in the case of e-mail.
For the realization of encrypted communication at telephone, user must have the right and method of deciding and using of encrypting algorithm. As in the case of e-mail, we can use that crypt program having the character in [0071].
The structure and movement is same as the case of e-mail. Only one point is different from the case of e-mail, it is that voice data is treated.

と変更します。

２．３　について、
請求項２は現在
2. A protected communication computer software according to claim 1, further comprising electronic mail transmitting means for transmitting the enciphered transmission anticipated file attached to an electronic mail, to the designated transmission destination.

ですが、書き換えるなら

2. A computer software, used in protected communication, according to claim 1, further comprising electronic mail transmitting means.
The file to be transmitted is encrypted before transmission.　After the encryption the encrypted data is attached to an electronic mail.
When the data is enciphered, the encrypting software and key are chosen using extended address book. [Fig.1], [Fig.13]
The designated transmission destination is compared with the e-mail address in the Fig.1, Fig.13.

ですが、これを請求項として残してもあまり意味がない。

３．　について
　すでに提出されている、請求項１の表現は、詳細の記述に於いて示されている内容を十分に表現できていないことは認めます。
　この技術の進歩性

１．通信の安全性を高めるには、強力な暗号ソフトが必要です。それを開発した人に経済的な保証を与えるには、購入した人が簡単に利用できることが必要です。
２．新しい理論に基づいた暗号アルゴリズムに関しては、購入した人が簡単に古いものと入れ替えることが出来なくてはなりません。
３．鍵の長さな安全性の要です。利用者が自由に設定できなくてはなりません。
４．強力な暗号化は、サーバーに関する権限を持たない人でも利用できなくてはなりません。
５．現在あるインターネット環境の中で実現できなくてはなりません。
６．解読技術の発達に対する抵抗力を持たなくてはなりません。

これらの課題を実現するものが、私の発明です。現在の設備で安全な通信を実現する方法であり、暗号技術の進歩を促す方法を現実的に提起するものであり、十分な進歩性があると信じています。

１．については、分離型暗号ソフトが私の提起する方法です。この形式では、購入した人は復号化部分の再配布が出来ないので、それぞれ暗号ソフトを購入することになります。暗号化ソフトの利用者分だけは、販売できることになります。
２．については、拡張型アドレス帳の項目を、新しいソフトに合わせて、新しい名前に書き換えるだけです。入れ替えは簡単です。アルゴリズム ID の必要もないので、発表と同時に使うことが出来ます。
３．については、鍵は利用者が自分で作ります。分離型暗号ソフトにはこの機能があるから簡単に作れます。好きななが っさの鍵を作って利用すればよいのです。安心できる長さの鍵が作れます。しかも自分で作り、第３者機関にその一部を登録する必要もないので安心して使えます。
４．については、技術は安く提供すべきです。メールアドレスを自分で発行出来るようにするには、お金がかかりすぎます。一つしかないメールアドレスで何とかすべきです。私の方法なら実現できます。
５．については、データパケットの構造を変えない範囲で、暗号通信を行えば新技術をすぐに利用できます。国 ID やアルゴリズム ID の設定などはパケットのヘッダー部分の変更となり、現存するサーバーを換えるまでは利用できません。現実性のある方法を提起しています。
６．については、多重暗号化がその答えです。１０段階ほどの、異なるアルゴリズムによる暗号化でとりあえず十分でしょう。必要なら、アルゴリズムと鍵のセットを何組でも記述したファイルを用意しておけば、暗号化の段数に制限は無くなります。

　これは、集合から順序対への概念の変換が与えてくれたものです。この変更が現在のインターネットの構造を変えることなく、強力で意味のある暗号化通信を実現してくれます。この方法は、暗号化ソフトの利用を促進し、その開発と販売を活性化させる力を持っています。
　旧来の技術を表している文献＜Ｄ１＞は、強固な暗号通信を実現すると言うよりは、本来の暗号化による秘密保持の目的に反している内容を示しています。また、文献＜Ｄ２＞は、インターネットの世界に新しいメールサーバーを設置しなくては実現できません。
　結果から見れば私の方法は単純です。現存するインターネットはすでに、base64 による変換で任意のデータを送信できる力を持っています。暗号化されたファイルをバイナリデータと考えれば、その扱いは簡単になります。あとは、暗号化ソフトの形[0071]と、それを組み込んだメーラーの構造[Fig.1], [Fig.13], [Fig.2], [Fig.3]を考えるだけです。
　発想の転換は後から見れば単純ですか、最初に思いつき、具体的なソフトとして実現し、最強の暗号化通信の環境を全ての人に提供するのは、簡単ではありません。
　このメーラーを使えば、通信傍受による盗聴に対する十分な対抗策となります。しかも現存するインターネットの世界の中で、利用者が自分のメーラーを変更するだけで済みます。しかも、最新の暗号化技術を簡単に利用できるようになり、暗号開発者に経済的恵みを与えてくれます。
　文献＜Ｄ１＞、文献＜Ｄ２＞の問題点と限界については別の場所で詳しく記述しますが、私が提案するこの方法は、特許として認められるのに十分な進歩性を持っていると確信しています。

３．１　文献＜Ｄ１＞との違いを説明します。
最初に、使用する暗号化プログラムですが、この暗号化プログラムは

[0098] According to the present embodiment, the commercial software for enciphering is produced having following structure:
[0099] (1) Having an enciphering key creation function, and freedom of distributing the key is authorized.
[0100] (2) The redistributing the enciphering software is authorized.
[0101] (3) Has a deciphering key creation function.
[0102] (4) The redistribution of the deciphering key can be prohibited.

の様な仕組みです。この構造によって、ユーザーが任意の長さの暗号化鍵を作成できます。もちろん、その暗号化鍵に対応した復号化鍵も作成できます。この自分で作成した暗号化鍵を、暗号化ソフトと共に、前もって通信する相手に送っておきます。利用者は、市販されている中の最強の暗号化ソフトを自由に選べます。暗号化ソフトの部分だけを送り、復号化ソフトの部分は送りません。暗号化ソフトを直接送るのでアルゴリズム ID は必要ありません。開発されたらすぐに利用できます。しかも、復号化部分は再配布できないので利用者は分離型ソフトを購入することになり、開発者がその苦労に値する経済的報酬を得ることができます。
文献＜Ｄ１＞では、暗号ソフトの形式については触れられていません。従来の公開鍵暗号、秘密鍵暗号方式の議論しかありません。この議論では、いろいろな暗号ソフトの研究開発を促進するのは無理です。鍵の分離だけではなく、暗号ソフトを分離し、暗号化ソフトと復号化ソフトに分けることが必要です。この形と拡張アドレスブックの組み合わせが、大衆が買ってくれる暗号ソフトの形です。開発者が儲かる暗号ソフトの形です。これが、暗号技術の更なる進歩を実現する鍵です。
私の表にあるのは、アルゴリズムではなく、相手から送られた暗号化ソフトと、相手から送られた暗号化鍵、自分で買った復号化ソフト、自分で買った復号化鍵です。暗号アルゴリズムを登録するとの表現は、暗号ソフトを分割し、ID なしで直接扱うことを無視しているからいえる表現です。従来技術にとらわれているから分離型暗号ソフトの性質を無視して、”encryption algorism” を “encryption software” と “deciphering software” の統一的な表現として使ってしまうのです。
関係が定められているのは、分離型暗号ソフトの暗号化ソフトと復号化ソフトです。
送信時には、宛先アドレスと暗号化ソフトが対応し、
受信時には、送信元アドレスと復号化ソフトが対応するのです。
この対応は、分離型暗号ソフトを使っている時の対応です。文献＜Ｄ１＞では、暗号ソフトを分離する考えがないからこそ、ID によって、暗号アルゴリズムを指定するとなっているのであり、分離しなければ、ソフト購入の経済的負担を受信者だけが負うようにはできません。暗号ソフトを分離しなければ送信者と受信者の双方が経済的負担をすることになります。これでは、暗号ソフトの有効性について意見が対立したときの解決が困難になります。
さらに、文献＜Ｄ１＞のテーブル１の問題ですが、このテーブル１の左端の欄は、送信先を意味していません。送信先となる人間が住んでいる国です。国民が２人以上住んでいる国では、送信先を、国にしたのではどの国民がその送られてきた情報の受取手なのかが分かりません。
送信先とは、情報の最終的な受け手を指し示す言葉であり、情報の受け手が住む国ではない。従って、文献＜Ｄ１＞の図１における国(country)は送信先ではない。つぎに、文献＜Ｄ１＞の暗号化プログラムの欄には、DES,RC5 の２種類の共通鍵暗号化方式の名前が全てに記入されている。これは、アルゴリズムに ID を付けて管理する方式の必然的結論です。アルゴリズム ID は暗号技術の発展を阻害します。もし、国からの出口のサーバーにだけ暗号化ソフトが置かれるにでは、購入される本数が、極端に少ないので開発者の儲けはありません。開発しても採用されない可能性もあります。暗号技術が発展する形態が必要です。
暗号化方式は、送信者と受信者の各順序対に対して決定されるべきであるというのが、私の主張であり、これは、私の図の（(a)Mr.Akiyma’s address book,と(b) Mr.ito’s address book,
における、X Company の部分を比較すればより明らかであろう。
(a)では、暗号化鍵は　KCax 暗号化ソフトは　Cax
(b) では、暗号化鍵は　KCix 暗号化ソフトは　Cix
となっている。

[0061] An enciphering method is determined corresponding to this ordered pair. When (X, Y) is set as a transmission from Mr. X to Mr. Y, the receiver, Mr. Y has the freedom to determine the enciphering method in this case as a general rule.
[0062] Further, the information transmitted from Mr. X to Mr. Y, is automatically enciphered with predetermined enciphering method chosen by Mr. Y.

[0063] The enciphering technique relied upon by Mr. Y does not necessarily be the technique that is relied upon by Mr. X. Therefore, when Mr. X receives information from Mr. Y, the enciphering method determined by the ordered pair (Y, X) is used. In other word, the transmitted information is enciphered by enciphering method pre-set by Mr. X regarding Mr. Y

からも分かるように、
私の方式では、暗号化ソフトの各欄には１種類の名前しか記入できない。暗号化ソフトの欄と復号化ソフトの欄には直接ソフト名が入る。したがって、アルゴリズム ID は使わない。最新のアルゴリズムを使うには ID が決まるのを待って入られない。解読技術が開発される前に利用できるシステムが必要である。ID を使ったのでは、解読技術が開発されるまで待っているようなものである。
暗号開発の発展には、自由な販売が不可欠である。ID はその妨げにしかならない。したがって、拡張アドレス帳は、ソフト名が直接記入できる方式でなくてはならない。
さらに、鍵の欄には長さの制限を記入するなど暗号を解読してくれといっているようなものである。鍵は利用者が直接作成し、自由に長さを決めるべきであり、鍵の項は自由に作成した自由な長さの鍵ファイル名を記入するようにしなくてはならない。長さ制限を記入するべきではない。
しかも、受信者が同じでも送信者が異なれば暗号化ソフトも異なるものを使用するのが、原則である。何しろ順序対語とに暗号方式を決めるのが根本的な考えだからである。多様な暗号を利用した方が解読しにくい。
多段階の暗号化の場合には、Fig.13 にあるように、別の欄にして、それらの適用される順序が明確になるように記述しなくてはならず、同一の欄に２つのファイル名を書き込んではならない。
暗号の強化には多重暗号化が不可欠であり、適用されるアルゴリズムの順序を表せる項目が必要だが、文献＜Ｄ１＞には記述がない。

[0066] This address book is expanded and added with four items (columns), enciphering key, enciphering software, deciphering key, and deciphering software. The address book is modified as shown in the example in FIG. 1(a).
[0067] The symbols used in the same figure (a), KCax, Cax, Kpxa, and Pxa are set as the following:

1 KCax: An enciphering key for enciphering communications from Mr. Akiyama to X Company. Cax: An enciphering software for enciphering communications from Mr. Akiyama to X Company. KPxa: A deciphering key for deciphering enciphered communications from X company to Mr. Akiyama. Pxa: A deciphering software for deciphering enciphered communications from X company to Mr. Akiyama.

から分かるとおり、
送信元データとは、実際にデータを送信する個々の人の識別記号を意味する。これは、送信者が住んでいる国のことではない。具体的にはそれぞれのメールアドレスである。
さらに大きな違いとしては、文献＜Ｄ１＞のテーブル１には、暗号化アルゴリズムの欄はあるが、復号化アルゴリズムの欄がない。私の図１には、同じ X Company に対して、暗号化に関する欄（ソフトウエアと鍵）が２つ、復号化に関する欄（ソフトウエアと鍵）２つが別に存在する。この４つの欄は順序対（a,x)と(x,a)を異なる２つのものと考えているからこそ、存在するのである。
暗号ソフトの販売、開発を盛んにするには、私の提案する分離型暗号ソフトが必要である。暗号での鍵を公開鍵と秘密鍵に分けて公開鍵暗号方式が提案されたが、これだけでは不十分である。鍵だけではなく暗号ソフトも２つに分離しなくてはならない。そして、暗号化部分を暗号化鍵と一緒に相手に送るようにすれば、拡張アドレス帳に登録するだけで、新しいアルゴリズムが簡単に利用できる。これならば、新しいソフトを開発して、販売し利益を上げられる。多重暗号化は、ソフトの種類を増やすことに貢献する。結果として安心してインターネットで通信が出来るようになる。
２人の間の通信を、集合｛a,x｝における通信と考えているのが古い考えであり、強力な暗号化を実現できない理由である。
そうではなく、２つの順序対（a,x)と(x,a)であると考えたことが、私の主張する新しい通信方法であり、順序対の概念が、通信ソフトの構造を決定する。順序対から分離型暗号ソフトの考えが生まれる。
この違いが分からないからこそ、同じテーブルの暗号化アルゴリズムの欄を送信元のいる国のデータと結びつけ、復号化に使えるかのように理解するのである。これは大きな誤解である。私の表の暗号化と復号化の欄が別に存在する理由は、同じ相手との通信でも送信と受信では、順序対が異なるので、当然異なるアルゴリズムが使われると考えているのである。
私の図１では、暗号化と復号化が、異なる記号で表現してあることを確認してください。
さらに、文献＜Ｄ１＞のテーブルでは、鍵の長さについて、多くの数値が記載されているが、私の図には、

そのようなものは一切無い。理由は、鍵は暗号の利用者が自由に作り、配布するものだから、そのような長さ制限は一切考えない。
以上の点から、表にはいろいろあるが、その内容は様々であり、私の図１と文献＜Ｄ１＞のテーブルは内容が全く異なるものであると主張する。
私の表は、暗号技術の発展を助けるが、文献＜Ｄ１＞の表は暗号技術の発展を妨げる。全く逆の働きをする表である。したがって、文献＜Ｄ１＞の表よりも進歩性を持っている。

(a) について、文献＜Ｄ１＞page 489 では、
The encryption algorithms used within SKR may also be freely chosen, and the choices are indicated as parameters within the T1 and T2 headers.
となっている。問題は多い、暗号化アルゴリズムは、本当に自由に選べるだろうか？
この方式では無理である。Ｔ１の中には、session key type (i.e. encryption algorithm -DES,IDEA,..)P498 が含まれている。これでは、新しい暗号化アルゴリズムをすぐには使用できない。新しい暗号アルゴリズムに対するパラメーターは誰が何時どのようにして決めるのであろうか？X 国と Y 国が対立することや、戦争をすることも多い世の中で、X 国が新しい暗号方式を開発したとする。しかも、このアルゴリズムなら X 国の政府にとっては簡単に解読できるが、Y 国の政府には解読できないとする。X 国は、この暗号化が Y 国に対して優位に立つために役立つと考える。この暗号化方式を世界標準にして、使いたいと思うであろう。X 国は、この新しいアルゴリズムのパラメーターを決めて使用を開始するように国際会議で主張する。Y 国は、その暗号の解読技術を発見するまでは、この X 国の提案に反対するであろう。Y 国が同意するのは、解読技術を身につけた時であろう。３年もすれば暗号解読技術は飛躍的に進歩する。Y 国が同意する頃には、新しかった暗号技術も昔のものになってしまう。
パラメータを自国内だけで使用するなら、同じパラメータを別の暗号化アルゴリズムに割り当てた国との暗号通信は不可能となる。
暗号解読技術の進歩、国際的な対立、の点から見て、パラメータで暗号アルゴリズムを決めるのは現実的な意味を持たない。
パラメータを決めるまでは新しいアルゴリズムは利用できない。発明者が勝手にアルゴリズムのパラメータを決めるならば新しい暗号方式を、狭い範囲に限れば、すぐに利用できるが、世界中での通信を考えると異なるアルゴリズムに同じパラメータが与えられてしまう。これでは通信できない。
暗号は利用する人が、必要とするだけの強さを持った暗号化アルゴリズムを利用できなければ意味がない。破られることが前提とされる暗号は意味がない。アルゴリズムにパラメータを付けたり、鍵の長さを制限すると、暗号化しても破られてしまうものしかできない。
破られない暗号化を実現する方法が必要である。
この後半部分であるが、
現存するインターネットで、このＴ１やＴ２を含むデータブロックは正しく機能するのだろうか？この構造は現存する、SMTP や POP3 などのメールサーバーでは全く機能しない。文献＜Ｄ１＞の実現には、世界的規模での新たな経費がかかる。

私の場合は、
[0071] A blank e-mail is created, listing only the address of the sender and the address of the destination, and the enciphered information is transmitted as an attachment file to this blank e-mail.
から分かるように、現存するインターネットに直接流せる。そして完全に機能する。理由は簡単である。バイナリーファイルを base64 で変換してメールサーバーを通せるからである。言い換えれば、どのような形式のファイルも添付ファイルとして送信できるという、インターネットの性質を利用しており、新しい内容のヘッダーをパケットに加えてはいない。
したがって、利用者が新しいメーラーを買えば、そのときから強力な暗号化の恩恵を受けられる。

比較すれば、文献＜Ｄ１＞では、新たなメールサーバーを世界的規模で再構築して、さらに暗号の利用者が自ら通信傍受を助けるような作業をしなくてはならない。
私の方法は、現在のインターネット環境のままで、利用者が欲しいだけの強度を持たせた暗号通信を提供し、通信の傍受を妨げるために利用できる。
最後に、どのような暗号化方式でも使えるようにするのは当然である。

文献＜Ｄ１＞では、鍵の長さの制限に配慮するあまり、現在のインターネットでは、意味を持たないものを提起している。私は今のままのインターネットで意味を持つ暗号通信の方法を提起している。これが、もっと

も大きな違いである。

（ｂ）暗号化アルゴリズムの決定権を誰が持つかについてですが、国の ID やアルゴリズムの ID を使うとなると、最新の暗号化アルゴリズムが発表されたとき、その新しいアルゴリズムの ID が決定されるまでは、その暗号化技術は利用できないことになる。さらに、国の ID は現在決まっているのだろうか？　私は日本の国 ID を知らない。この国 ID の一覧はどこで入手できるのだろうか？　新しい国が出来たときその国の ID は何時誰が決めるのか？

この観点から見て、文献＜Ｄ１＞の方法は、最新の暗号化アルゴリズムの利用を妨げる性質を持っている。このような、国 ID、アルゴリズム ID とは無関係に、暗号化アルゴリズムを採用する権利は各個人が持つべきである。

現在のネットワークで、今すぐ機能するようにするためには、現在すでに存在している識別記号すなわち、メールアドレスのみで暗号化アルゴリズムを決定できなくてはならない。その意味で文献＜Ｄ１＞の方法は実現するのが困難であり、優れた技術とはいえない。

暗号を利用するのに ID はいらない。暗号化ソフトの名前を登録し、幾つかの引数と共に呼び出せばよい。メーラーにこの機能を持たせるのは簡単である。分離型ソフトなら、ID 無しにすぐに利用できる。

文献＜Ｄ１＞の記述は、現存しない２つの ID が必要であるとの記述であり、私の主張は現存する ID（メールアドレス）を１つだけ利用するとの記述であり、現在のインターネットで利用可能か、利用不可能かという観点から見れば、全く異なる性質を表現している記述でありこれらを同一視する事は内容を誤解しているといえる。

なお、分離型暗号ソフトの作成は簡単であり、近日中にインターネットで入手できるようになる。

（ｃ）この議論は、暗号化した通信では一般的な事柄である。復号化しなければ受信した内容を理解できないのだから。コンピュータを使う以上自動的に解読されることは当然であり、この点は文献＜Ｄ１＞の進歩性を示すものとは思えない。この点が同一だからといて特許が否定されるのでは、これからは、受信したデータを紙と鉛筆で解読するような暗号化しか認められなくなってしまう。

自動解読出来ることだけが独立して、特許の内容として主張されるべきではない。もちろん、文献＜Ｄ１＞において、暗号文が自動的に復号化される場合は、国ごとにすでに定められている暗号化アルゴリズムの中から、ヘッダー情報Ｔ１を使って使用するものを選ぶのであり、この点で暗号技術の発展を妨げる要素を含んでいる。最新技術はすぐに利用できなくては意味がない、国が検討している間に、解読技術が発表されてしまいます。文献＜Ｄ１＞の主な主張は暗号化鍵の扱い方にある。この暗号鍵の扱いが特許の主な内容である。

アルゴリズムに関する情報を、パケットに含ませることは、暗号化の強度を弱めることであり、本来の目的に矛盾している。解読に利用できる情報は減らすべきである。

文献＜Ｄ１＞の内容では、鍵の長さは国が決める場合を主に考えている。私はこれとは全く反対に、それぞれの利用者が、自分で自由（長さを含めて）に暗号化鍵を作成することを主張している。

私は、順序対の概念に従った、拡張版のアドレス帳と、４つの性質を持つ暗号ソフトを利用した、自動解読を主張しているのである。

変な ID は必要ない。国やプロトコルの ID はあると有害。国と国の利益が衝突すると
ID は決まらない。Ｃ国は自分だけが解読できるなら ID をきめて使いたいが、Ｒ国は解読できないので使わせたくない。Ｃ国政府は、利害が対立するＲ国に暗号解読方法を教えるのだろうか。妨害するには同じ ID を別の方式に与える。

３．２

確かに、現在の請求項１は、そうなっています。しかし、詳細部分を見れば、

１．拡張アドレス帳の各項目に登録されるのが暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵であり、暗号アルゴリズムの ID ではない。、
２．アルゴリズム ID は使わないで、分離型の暗号ソフトを使う。（ソフトの名前を登録する）
３．多重暗号化が可能。

４．暗号化鍵は利用者が自由に作成し配布する。
５．インターネットのサーバーの変更は必要ない。
６．インターネットに流すパケットのヘッダー部分の変更をしない。
７．利用者が拡張アドレス帳を備えた新しいメーラーと分離型暗号ソフトを買えばすぐに利用できる。
８．ユーザーのパソコンから出る時点でデータは暗号化されている。
９．ユーザーのパソコンに入ってから復号化される。

などの違いがあります。請求項１は、これらの違いを鮮明にする形で書き直されなくてはなりません。

この違いから、文献＜Ｄ１＞は現在のインターネットですぐには実現できないのに、私の方法は明日にでも、現在のインターネット環境のままで実現できるという違いが有ます。
この実現についての違いは、小さく思える文献＜Ｄ１＞との違いが本質的な、大きな違いでることを意味しています。拡張アドレス帳を持っている新しいメーラーも、分離型暗号ソフトも、明日にでもインターネットを通じて販売できます。

３．３
本発明の課題を整理すれば、

次の条件を満たす普通の人に、強力な暗号通信の手段を与えることです。

１．メールアドレスを１つだけもっている。
２．サーバーの管理権を持たない。
３．データパケットの構造を換える権利を持たない。
４．新しいメールアドレスを自分で発行することが出来ない。
５．インターネットのプロトコルを変更する権利を持たない。

まとめれば、暗号の利用者に対しては、現在のインターネットのままで、サーバーを導入することも変更することもなく、与えられたメールアドレスのままで、望むだけの強さの暗号化通信の方法を安く提供すること。さらに、暗号開発者には経済的な報酬が与えられる仕組みも提供することです。

これは、現在のインターネットでバイナリファイルの転送が可能であることを利用すれば、次の方法で解決できます。

１．分離型暗号ソフトを使う。
２．拡張アドレス帳を使う。
３．多重暗号化をする。

この条件を満たすメーラーと、暗号ソフトを作ればよい。
この考えは、２つのものからなる集合｛ｘ、ｙ｝として通信をとらえるのではなく、通信は２つの順序対（ｘ、ｙ）と（ｙ、ｘ）からなるものとして考えることから導かれます。
拡張されたアドレス帳が順序対の概念を表現して、この概念に対応するソフトが分離型暗号ソフトです。この二つがこの発明の特徴です。
これによって、任意の強度で、自由に暗号化できるようになります。
他の多くのものが集合｛ｘ、ｙ｝を基本にして対応テーブルを構成しています。これでは強力な暗号化の実現はできません。

３．４
請求項１の書き方が、自動的に暗号化されるという点に表現が偏り、詳細部分に書いてある特徴が表現出来ていないのですが、詳細部分で展開してある内容は以下の点で文献＜Ｄ１＞とは大きく異なります。

文献＜Ｄ１＞では、アルゴリズムアルゴリズムにIDを付けるまで利用できません。私の方式ではIDは必要ありません。新しいアルゴリズムは分離型暗号ソフトとして作れば、すぐに販売できます。購入すればすぐに利用できます。

文献＜Ｄ１＞では、使用する暗号ソフトが暗号化部分と復号化部分に分離されていない。暗号ソフトの有効性について意見が対立したときに、問題が起きる。役に立たないと思う暗号ソフトを買う人はいないから。私の方法なら、分離型暗号ソフトなら、復号化部分を手にした人がお金を払う。暗号化部分は無料で再配布されるから、暗号化部分だけを使う人はお金を払わない。自分の信用する方式だけを購入すればよい。

文献＜Ｄ１＞では、暗号化アルゴリズムのIDを含むデータを送信しなくてはならない、それに対して、私の発明では、通信傍受と第３者による暗号解読を助けることにしかならない暗号化アルゴリズムに関する情報を送信するパケットの中に含めることはしない。

文献＜Ｄ１＞では、与えられたテーブルを使って、国ごとに決められたアルゴリズムの中から選ぶ形であり、通信する人が自分だけの対応テーブルを作るわけではない。
私の方法の、図１は暗号の利用者が自分だけの対応テーブルを作ることを意味している。
私の場合は幾つかの許されたアルゴリズムからIDを使って国によって与えられている復号化アルゴリズムの中から選んでそれを適用するのではない。図１の中に、一つだけがすでに書き込まれているので、新たに付けたヘッダー情報を使うことはない。

文献＜Ｄ１＞では、テーブルの管理に問題がある。国でまとめて暗号化アルゴリズムを管理し、このテーブルが存在するサーバーを通るときに暗号化されるとすれば、個人のパソコンからこのサーバーまでの間は暗号化できないのであろうか？暗号化されないとすれば、サーバーに着く前に盗聴されてしまう。個人のパソコンからサーバーまでの間を個人が勝手に暗号化するのでは、鍵の長さを規制したりする意味がない。個人が法律通りに暗号化しているか否かはそう簡単には分からない。１６進表現した３２ビット整数からなる乱数データである。と主張するバイナリデータが、何かを違法に暗号化したものか否かは全ての暗号を解読できなくては判断できない。解読終了までデータが送信されないのでは、インターネットは停止してしまう。
個人で管理するとすれば、テーブルの更新を忘れた時、通信が出来なくなる。新しいアルゴリズムのIDが決まったら各個人は、新しい暗号ソフトを購入しなくてはならないのであろうか？購入出来ない人も多くIDの登録はしても、ソフトが存在しないことになり、機能しなくなる。
私の方法では、自分が使う範囲内で管理すればよい。知り合いの分だけ管理する。いやなら登録しないと相手に連絡すればよい。個人のパソコンの中で暗号化され、パソコンから外に出る時点ではパケットのデータ部分は暗号化されている。解読は、宛先となっている人が持っているのパソコン中である。ネットワークに流すのは全て暗号化すべきである。

暗号化と復号化のアルゴリズムを正しく対応づけることは必要なことであり、それ自体が特許の内容としてはふさわしくないのは当然である。対応づける時の表の構成が、通信は２つの順序対であるとの考えを元に構成された対応表、すなわち拡張アドレスブックであり、この形式に対応する暗号ソフトが分離型の暗号ソフトであり、拡張アドレス帳と、この分離型暗号ソフトの組み合わせは、暗号の利用を簡単にし、全ての通信が十分強い暗号化をしてから送信される社会を実現し、さらに暗号開発に対する経済的保証と与えることが出来る革新的な形態であると信ずる。特許の内容として相応しいものだと確信しています。
同じテーブルという言葉で表現しても、内容がまったく異なるテーブルである。文献＜Ｄ１＞のテーブルは、真に有効な暗号の利用を妨げ、暗号技術の発展を阻止する内容である。私のテーブルは、新しいアルゴリズムをすぐに利用で知る基盤と、暗号発展の経済的基盤を提供するものでめ。アルゴリズムの管理の仕方がまったく異なる。同一視することに疑問がある。

３．５
この結論は、
請求項１は、表現が不適切ではあるが、記述された詳細部分では

順序対の概念から導かれた、図１の対応表を利用し、さらに、そこに記載されている暗号ソフトの構造が、利用者に対する暗号ソフトの利用について自由度を大幅に拡大しており、その結果、いかなる通信傍受に対しても対抗できる自動的な多重暗号化の方法を、現在のインターネットのなかで、実現する方法を提起している

ものである。
　さらに、分離型暗号ソフトは、開発者が経済的恩恵を十分受けられる基礎を提供している。安全な通信のためには、暗号技術の発展が不可欠であり、研究者の努力に対する経済的保証を与える仕組みが必要である。これを可能にする方式を提唱している。
　利用者は､新しいメーラーと､分離型暗号ソフトを買えば良い｡これだけで強力な暗号化通信を享受できる。

ことが、示されている。ということの認識を欠いている。

　公開鍵暗号では、公開鍵と秘密鍵に分離した。私は、暗号ソフトを、暗号化するときに使う部分と、復号化するときに使う部分に分けることを提唱している。登録も削除の簡単で、３年くらい立つと暗号解読の技術が大幅に進むことに合わせて簡単に更新できる。多重暗号化と鍵の長さを自由に設定できる。これにより安心して情報を伝達できるようにしています。
　したがって、集合｛a,b｝から、２つの順序対(a,b)と(b,a)に変えたことが単純ではあるが十分な進歩性を持った方法であると確信している。実際、この方法が一般的になれば、現在の通信傍受はきわめて困難となる。

　また、次のような暗号化の技術も可能である。

１．原文　　IBM　暗号化して　HAL を得る。
２．HAL　を 第２の平文
　　Happy A New Year.
　　Alice is my friend.
　　Love to Alice is forever.
　に変える。

　このときに人工知能を使って第２の平文をつくる。

３．この第２の平分
　　Happy A New Year.
　　Alice is my friend.
　　Love to Alice is forever.
　を送信する。
４．先頭の文字を取り、HAL を得る。
５．復号化して　IBM を得る。

　もちろん、１の暗号化アルゴリズムは秘密にされる。５も秘密のアルゴリズムを利用する。
　このとき、第２の平文を送信することは、平文を送信したことになるのだろうかそれとも暗号文を送信したことになるのだろうか？
　第２の平文を傍受した人は、暗号文を傍受したと考えるのか、それとも平文を傍受したと考えるのだろうか？
　この、第２の平文を作成する人工知能が完成し利用され始めると、現在の通信傍受が意味を持たなくなる。なぜなら、全ての平文による通信を傍受し、全ての暗号化アルゴリズムと
全ての暗号化鍵の可能性を考えて解読作業をしなくてならなくなるからである。こうなると傍受しても解読不能となる。

３．６
　順序対の概念を暗号通信に利用するとの観点は、十分な進歩性を持っていると考えます。
暗号アルゴリズムを、２つに分けて、暗号化ソフトと復号化ソフトに完全に分離し、これを使って、順序対の概念に適応した通信を実現するという発想は、当業者が当然にして辿り着くものではない、と考えます。
　拡張されたアドレス帳の構造は、受信者が自分に宛てて送られてくる情報を自由に、もちろんアルゴリズムに関する何の制限もなく、鍵の長さに関する何の制限もなく、暗号化できるようにしてくれる。
　これは文献＜Ｄ１＞のテーブル１では、まったく不可能である。順序対の概念こそが、暗号化に関する自由を与えてくれるのである。従って、私の方法は、進歩性を十分持っていると考えます。

３．７
文献＜Ｄ２＞　について
column 10,line 33 - column 11, line 27

＜反論＞
表の形は似ているが、この表の存在位置と有効性に疑問がある。現在スパムメールを防ぐのに、プロバイダーから与えられたアドレスの他に、ウエッブメールを使って、アドレスを複雑なものや、タイプのことなるものを用意し、使い分けるのは普通に行われていることである。
似ているのは表の列の数だけである。しかも、多重暗号化に関係のない基本形のみが列の数に関してのみ似ている。機能も使い方もまったく異なる。表形式だから新規制がないと言われると困るが、大切なのは表の項目に何を設定し、それをどの様に機能させるかという点である。
私の表は、ユーザーID、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵であり、これによって、順序ついに対応した暗号通信を実現する。さらに、多重暗号化も可能にし、意味ある暗号通信を実現する。もちろん必要なのは分離型暗号ソフトである。
単に迷惑メールを防ぐだけなら、無料のメールアドレスを沢山持って、しかも全て複雑なものにしておけばよい。しかしこれも、プロバイダーからアドレスが大量に流出し、犯罪的な形で販売されるとすぐにスパムメールが来るようになる。もちろん新しい無用メールアドレスをとり、友人に知らせればよい。
この方法の欠点は、個人または、会社で特殊な PCAHOST206 を所有または管理する必要があり、この PCAHOST206 はメールを受信したりアドレスを変換したりする特殊なサーバーである。メールアドレスの変更可能なサーバーを運用するのは個人では経済的に無理がある。ある会社で利用する場合、固定アドレスを取得し、サーバーを運営し、各ユーザーに幾種類かのアドレスを与えることも可能であろう。その結果、その会社の誰が暗号通信をしたかは隠せてもその会社から暗号通信が行われたことは隠せない。盗聴のターゲットが個人に絞られるのかそれとも会社に絞られるのかの違いしかない。盗聴者は会社の人数分の通信を解読すれば済むので、大した手間ではない。必要なら、社員の人数分だけ解読用のコンピュータを用意すればよい。この方法は暗号強度を特に高める要素を持たないので、情報を防衛する方法としては意味をなさない。
暗号化されないデータをインターネットに流すことは、裸で歩くのと同じである。洋服に当たるのが暗号化である。みんなが洋服を着るのと同じように、暗号化の着物を着せてデータをインターネットにながすべきであり、全員が暗号化するようにすれはよいのであり、誰のデータが暗号化されたかを知っても意味のない世界がすぐに来る。全員が暗号化されたデータを流すようになる。
普通の人は、数個のメールアドレスを使うことが多い。このとき、スパムメール対策でもっとも有効なのは、アドレス帳に登録されていない人からのメールは全て削除することである。メールサーバーには来ていても、自分のコンピューターにダウンロードする過程で、発信元のアドレスが解析された時点でアドレス帳とデータ比較をして、削除するか最後までダウンロードするかを決めるようにすればよい。アドレス帳に登録されていないが本当にメールを受け取ってもらいたい人は、前もって電話か手紙で、自動的に削除する機能を OFF にしておいてもらえばよい。

column 14,line 55 - column 16, line 16

＜反論＞
ネットワークでの傍聴が可能な人に対して、チャンネルの切り替えは意味を持たない。傍聴する人や組織はあらゆる通信チャンネルを傍受できると考えるべきである。
さらに、現状では、普通の電子メール使用者はプロバイダーから与えられたアドレスを持っている。これらはプロバイダーから与えられたものであり、自由に変更することは出来ない。ましてや、これを通信の途中で切り替えることは出来ない。
しかも、電子メールの良さは、送信者は自分の都合の良い時間にパソコンのスイッチを入れメールを作成し送信後パソコンの電源を切る。送信されたメールは SMTP サーバーを経由してプロバイダーのメールボックスに入る。受信者は自分の都合の良い時間にパソコンの電源を入れ、プロバイダーのサーバーのメールボックスにあるメールを取り出す。
普通パソコンは電源が切れている。メールはプロバイダーの POP3 サーバーにある。
直接ユーザー（Ａ）とユーザー（Ｂ）のパソコンが通信を行うことはない。

そこで、問題となるのは、
（A）の通信相手Bの UCDB の存在する場所はどこか？

１．（B）のパソコンの中なら、電源が切れているときは、電話して電源を入れてもらわなくてはならない。電子メールの本質、相手が自分の都合の良いときに自分のパソコン電源を入れてメールを取り出す。相手からの夜中の電話に起こされることもない。国際間のメールが一般手にな現在、不便きわまりない。また、電源を入れたままのパソコンはハッカーの攻撃対象になる。不安が増すだけである。
電話して電源を入れてもらうなら、その電話で話をしてしまえばよい。もし電話の盗聴が心配なら、その盗聴を防げばよい。

２．プロバイダーのサーバーの中か？、いまの SMTP サーバーや、POP3 サーバーはこの機能を持っていない。新しいサーバーがこの機能を持ったとして、世界中のプロバイダーにこの各個人のデータベースを保存すると、個人の友好関係が全て他の人に把握されてしまう。これは、個人のプライバシーの崩壊につながる。

３．メールサーバーを立てた会社ごとにあるのか。会社の誰からのメールかを隠しても意味がない。個人の顧客は、この変なサーバーに対応していないので、作っても効果はない。ハッキングの目標を作り出すだけである。

通信内容を安全に送るには、現在のインターネット環境のままで十分である。ただ、拡張アドレス帳のあるメーラーと、分離型暗号ソフトを使えばよい。

column 17,line 62 - column 19, line 14

＜反論＞
チャンネルの切り替えに何の意味があるのだろうか？電話してくる相手もただ一つの電話番号しか持っていない。普通の人は、電話会社から与えられたただ一つの電話番号しか持っていない。使用する回線も変更できない。この条件の中で実現できるものを考えるべきである。
公衆電話網の中のユーザー権限で出来ることをはっきりさせると暗号化の形式がいかなるものであるべきかとうい課題の答えが見えてくる。
出来ることは、デジタル化された音声の代わりに、デジタル化された音声を同じビット数のデータに暗号化して、送信することである。利用者が必要とするだけの強度を与え、暗号ソフト制作者の利益になるものを提案する必要がある。
利用するのは、分離型暗号ソフトと拡張アドレス帳である。これによって、パケットのなかにある、音声信号部分を暗号化して送れる。多重暗号化はユーザーが必要とするだけの、強度を与えてくれる。
送信時には、相手の電話番号から、暗号化ソフトと暗号化鍵が決まり、デジタル化された音声データを暗号化する。
受信時には電話は全て受け入れるが、復号化が必要か否かが拡張電話番号表を使って決定される。もちろん復号化のアルゴリズム、復号化鍵も同時にきまる。
しかも、使用される分離型暗号ソフトは、暗号制作者の経済的利益になり、経済面から開発に貢献できる。

チャンネルの切り替えが出来る人は少ない。電話番号を１つだけ持っている人は多い。このチャンネルの切り替えが出来ない人に暗号化した通信を提供するとことができるのが私の方法である。

３．７
文献＜Ｄ３＞（最後の文献について）
EP0702477A2 は、方向性としては、意味のある暗号化を実現しようとするものであると思います。その実現はサーバー機能を持ったソフトウエアとして実現されます。
この方式を実現するには世界中のサーバーソフトを変更しなくてはなりません。
私の方式は、現在のサーバーの機能をそのまま利用していますので、各ユーザーが自分のメーラーを変更すればそれで済みます。

普通のユーザーにとってプロバイダーのサーバーのソフトを変更するのは不可能です。規格に合わないヘッダーを持ったパケットをインターネットに流すことも出来ません。
従来技術としての EP0702477A2 は、技術としては優れていても、ネットワークのエンドユーザーが自由に利用することができない技術なのです。
したがって、その機能を自由に利用できるのはどの立場の人間かという点において、最大の違いがあります。

文献＜Ｄ３＞（sun microsystems inc)の各部分について

page 4, line 42 - page 5, line 26

＜反論＞
参照テーブルを、サーバーに置くことは、問題が多い。Ａの通信相手は何人いるのだろうか、個人なら、１０００人分くらい置けば良いが、そのネットワークの中に世界中の人を相手に商売をしている会社が幾つか有ったなら、サーバーはこの会社のための参照テーブルをどの程度の大きさで確保するのであろうか？この大きさは世界中の人間の数と多重度と会社数の積で求まる。さらに問題がある、顧客データは大事な財産である。それをプロバイダーに預けることは、顧客情報の漏洩を引き起こし、会社の崩壊につながる。サーバーを借りたとしても顧客情報は会社内のコンピュータに置くしかない。この観点から見ても、この特許は方法としても、利用できるものではない。
その会社が自前のサーバーを建てると、使用できるのは与えられた IP アドレスであり、その IP からその会社が特定できる。トンネルの効果は半減する。
問題は参照テーブルの構造である。暗号の利用者は、情報の漏洩を防ぐためにどのような構造の参照テーブルを利用すべきなのか、が問題であり、私は拡張型アドレス帳としてその構造を明らかにしている。しかも、登録されるのは分離型暗号ソフトの各構成要素である。暗号選択の自由と鍵生成の自由さらに、多重暗号化に対応できる参照テーブル形が必要とされるのに、この特許では、それが示されていない。ネットワーク管理者の立場からの提案だからである。ここでの課題とすべきは、ネットワークに対する権限を持たない単なるユーザーに満足できる暗号化方法を提供することである。
一般のメールユーザーはホスト宛のメールを出さない。相手のメールアドレスに向けてメールを送信する。これが、相手のホストコンピュータに直接届くことはない。プロバイダーのメールボックスに入り、それを取り出すのである。ローカルネットワークのサーバーに関する権限は持っていない。
ましてや、ネットワーク間の暗号アルゴリズムの決定権は持っていない。管理者が直接の決定権を持っているとも思えない。理由は一つのローカルネットワークの管理者だけで新しい暗号アルゴリズムの採用を決めても、他のネットワークの管理者がその使用を認めない場合が十分考えられる。さらにそのローカルネットワークに属するユーザーがその暗号化アルゴリズムと鍵の長さについて十分だと思わないこともある
また、パケットが２１０から、２２０まで届く間に傍受されることも考えられる。完全に暗号化した後で公衆回線にデータパケットを送るべきであり、暗号化しないで送ったパケットを途中で暗号化するのは、平文と暗号文のセットを傍受者に与えることになる。これによって暗号文だけが傍受されたときの解読が簡単になる。
この観点では、２５０までが、ユーザーのパソコンで行われるべきであり、各ユーザーは全てネットワーク管理者の権限を持っていなくてはならない。このときは、各ユーザーが勝手に暗号化の参照テーブルを書き換えることになり、大きな混乱をもたらす。
どのアルゴリズムかは各個人が自由に判断するべきであり、ネットワーク全体での使用は解読されやすくなし、多重化や順序を各個人が変更することが出来なくなる。ネットワーク全体での利用は暗号の強度を下げる。暗号ソフトはあくまでも個人単位で、ばらばらに利用すべきである。
さらに、暗号ソフトの著作権と再配布に関して問題が生ずる。
“ＡからＢ”と“ＢからＡ”は区別されるべきである。
ネットワーク全体で管理するのはネットワーク管理者による盗聴と解読の可能性を生む。
管理者が何時悪人に変化するかは不明である。とても危険なやり方である。
復号化鍵は自分だけが持っているのがよい。ネットワークを通じて送ってはならない。
送ることは、暗号を解読しやすくするだけである。
普通の利用者はローカルネットワークの管理権を持たない。従って、ネットワークからネットワークへの暗号化は普通の人にとっては意味を持たない。
ネットワーク間で暗号化を統一すれば暗号が解きやすくなる。同一アルゴリズムで暗号化してある暗文が増えるからデータが増えて解読しやすくなる、例文が増えると統計的な処理がしやすくなる。ネットワーク内に平文を提供してくれる人がいればそれと暗文を比較して解読処理がやりやすくなる。
そもそも、２つのローカルネットワーク間の暗号化方式と鍵の長さを決める権利はどちらのローカルネッ

トワークの管理者が持つのだろうか、自分だけが解読方法を知っていて相手の管理者が解読できない暗号化アルゴリズムと鍵の長さを選ぼうとするのが当然である。
　特にローカルネットワークが一つの国であるときは国の利益が反映されるから。両方の管理者が合意できるのは、解きやすい暗号になってしまうであろう。暗号開発者と管理者の経済的癒着は当然生じる。３年後には廃れている暗号化が継続され、暗号としての意味を持っている新しいアルゴリズムは採用されない。
　しかも、サーバーを変更する権限のないものにとっては、パケットのヘッダー部分の内容を勝手に変更することは出来ない。現在与えられている TCP/IP のプロトコルに従ったヘッダーを付けなければ通信は出来ない。
　普通の、権限のないユーザーに、必要なだけの強さを持った暗号化の方法を提供することが必要である。これを実現するのが、分離型暗号ソフトと拡張アドレス帳である。これは、通信は２つの順序対であるとの考えから生まれる。
　普通は、送信者と受信者は、プロバイダーから与えられたアドレスしか持っていない人間や、ローカルサーバーを立ち上げることがでいない人にとっては、“誰から誰への通信か”は隠すことが出来ない。
　これを隠す必要はない。内容さえ分からなくすれば十分だと考える。
連絡と調整の手間な必要だが、分離型暗号ソフトの使用は、最新最強のアルゴリズムをすぐに利用で来る点において、その欠点を補ってくれる。
　この手続きを忘れるような内容のものは、暗号化の必要はない。

page 6, line 15 - 24

＜反論＞
　自動的な復号化には、どこかに鍵と復号化プログラムが無くてはならない。問題は、そのアルゴリズムの採用と、鍵の長さの決定、異なるアルゴリズムでの多重暗号化による強化の権限を誰が持っているかである。
　個人が持つなら、参照テーブルの存在場所が、個人のコンピュータの中に有るべきである。
個人ならば、どんなに大きくて複雑でもたかが知れている。計算がたいへんなら新しい PC を買えばよい。
　つぎに、個人は、新しいヘッダーを付加えることが出来ない。通信経路を変えると言っても、実際には通信経路はそんなに多くはない。海底ケーブル、衛星通信、くらいであり、この通信を押さえることは経済力が有れば簡単です。自前の通信衛星と原子力潜水艦を持てば可能となる。さらにネットワーク管理者から情報を得れば、利用者が知らないうちに暗号が無意味になってしまう。
　管理者が、暗号化についての情報を売り渡すのは別に不思議ではない。管理者が鍵の長さを短くするのは不思議ではない、サーバーにかかる復号化のための計算負荷を減らしたいと思うのは管理者として当然である。参照テーブルも小さくしたいであろう、時間が節約できるから。しかし多様な暗号化が有り、多重化が可能であってこそ解きにくくなるのだから参照テーブルは複雑で大きな方がよい。管理者の立場と暗号の強化を求める立場は相反する。
　管理者が新たな暗号化アルゴリズムを採用することに積極的だとは思えない。参照テーブルの更新がサーバーに対する新たな負荷となるからです。
　自分で暗号化について決定権のある個人が、自分の使う暗号についての情報を売り渡すことはないし、鍵の長さを短くすることもない。
　（４４０）の部分は形が決まっていて、この部分が多重暗号化に対応できているか否かであるが、鍵が複数有ることは記述されていない。ましてやその順序の記述されていない。
　さらに、XOR 方式のように鍵そのものが大きなファイルであるような場合は、この方式では対応できない。ヘッダー部分が５Mバイトあるようなパケットは送るのが困難である。従って XOR 方式には対応できない。ファイル名を送っても意味が無い。同じ名前で中身が異なるファイルが存在するのだから。
　簡単で高速な XOR 方式が使えないのが大きな欠点である。

以上の部分が私の考えた反論でした。

７．２　弁理士の反論案（２００４年１２月６日）

<u>庁指令に対する応答案</u>

　宇山様の最初の特許出願（特願２０００－３８８９２１）の請求項を再度検討いたしましたが、出願

時の請求項のままですと、どのようなことをしたいのかは記載されているものの、表現が曖昧であるため、許可される可能性は低いものと思われます。

例えば、請求項１においては、暗号を送信者と受信者の順序対に対して決める、とありますが、それが具体的にどのようなソフトウェア上の処理となるのかは記載されておりません。また、暗号化方式を受信者が前もって自由に指定する、ということは、送信者と受信者との間の取り決めであり、ソフトウェアにて行われる技術的処理（手段）ではないため、特許として権利化することは難しいと思われます。

請求項におきましては、目的を実現するための機能（技術的処理）を具体的に書く必要があります。そこで、本出願におきましては、提出いたしました請求項に修正したものであります。しかし、請求項の書き直しは行いましたが、実質的には宇山様の発明を適切に捕らえているものと考えております。

請求項１

- 参照テーブルを備えたメモリを有し、そのメモリには、送信先データと暗号化プログラム、及び送信元データと復合化プログラムとの関係が予め対応付けられた参照テーブルを有する。
- 参照テーブルには複数種の暗号化プログラムと復合化プログラムが記憶されている。
- 暗号化指示の入力があった時に暗号化プログラムに基づいて送信予定ファイルを自動的に暗号化する。暗号化プログラムは、参照テーブルを送信先データを鍵として参照することによって自動的に特定される。暗号化プログラムは、情報処理装置のメモリから自動的に読み出される。
- 復合化指示があった時に、復合化プログラムに基づいて受信した暗号ファイルを自動的に複合化する。復合化プログラムは、参照テーブルを送信元データを鍵として参照することによって自動的に特定される。復合化プログラムは、情報処理装置のメモリから自動的に読み出される。

このような具体的な機能によって、本発明の要部である、順序対に基づいて対応付けされた暗号化プログラムと復合化プログラムとを自動的に実行する、ということをソフトウェアの処理として表しております。上記のように、暗号化プログラムを受信者が選択できるということは、特許の対象とはならないものと考えられるため、送信者と受信者との順序対に基づき暗号化並びに複合化方式（プログラム）が特定されるということに焦点を絞ったものであります。

請求項２

- 暗号化した送信予定ファイルをメール添付として指定された送信先に送る送信手段を有する。

暗号化されたファイルは電子メールの添付ファイルとして送信されることを特定した請求項です。このように電子メールの添付ファイルとして暗号化されたファイルの送受信を行うため、既存のインターネット並びにメールソフトを用いて暗号化並びに復号化を行うことを可能としております。

請求項３

- 参照テーブルは、複数の暗号化プログラムを備えることが可能とされ、一つ送信先に対応する適応順番を備える。適応順番に基づき複数の暗号化プログラムを用いて、送信予定ファイルを多重暗号化する。
- 参照テーブルは、複数の復合化プログラムを備えることが可能とされ、一つの送信元に対応する適応順番を備える。適応順番に基づき複数の複合化プログラムを用いて、暗号化された受信ファイルを複合化する。

複数の暗号化プログラム（復合化プログラム）をもちいて多重暗号化をおこない、その暗号化（復合化）プログラムの使用順番を特定する適用順番データに基づいて多重暗号化（復合化）を行うことを記載した請求項です。そして、その適用順番データは参照テーブルに記憶されている。

請求項４

- 参照テーブルは、送信先データまたは送信元データ毎に、電子メールと本文と電子メール添付ファイルとのそれぞれ別個に複数の暗号化プログラムまたは複合化プログラムを対応付けすることが可能となっている。

この請求項は、メール本文と添付ファイルとをそれぞれ独立して暗号化及び復号化を行うようにできることを表しております。これにより、メール本文と添付ファイルとを内容の重要度や容量の大小等を考慮して多重暗号化の有無並びにその重合度を異ならせたり、必要に合わせて個別に行うことができるものです。

請求項５

- 適用順データは、予め定められた規則に基づき適宜自動変更することが可能である。

このような構成とすることで、定められた複数の暗号化（復号化）プログラムを用いていても、その適用順番を自動的に変更させることにより第３者に解読されてしまうリスクを減らすものです。

請求項６
- 暗号化プログラムと複合化プログラムとを入力する手段を有する。
- 発信先データと暗号化プログラムとを対応付けした暗号化参照データを記憶する手段を有する。
- 発信元データと復号化プログラムとを対応付けした復合化参照データを記憶する手段を有する。
- 通話にかかる発信先データと暗号化参照データに基づき暗号化プログラムを特定し、その暗号化プログラムを使用して音声信号を暗号化して送信する手段を有する。
- 通話にかかる発信元データと復合化参照データに基づき復合化プログラムを特定し、その復合化プログラムを使用して音声信号を復合化して音声出力する手段を有する。

請求項１に記載の暗号化方法を電話に適用した場合の請求項です。

請求項７
- 参照テーブルは、複数の暗号化プログラムを備えることが可能とされ、一つの発信先に対応する適応順番を備える。適応順番に基づき複数の暗号化プログラムを用いて、音声信号を多重暗号化する。
- 参照テーブルは、複数の複合化プログラムを備えることが可能とされ、一つの発信元に対応する適応順番を備える。適応順番に基づき複数の複合化プログラムを用いて、暗号化された音声信号を複合化する。

請求項３に記載の暗号化方法を電話に適用した場合の請求項です。

引用例との比較
請求項１について
本願発明においては、送信者と受信者との間にて直接暗号化プログラム及び複合化プログラムとを特定するものであります。そして、どの暗号化プログラム又は復号化プログラムを用いるかについては、送信者と受信者との順序対に対応付けされて、メモリ内に記憶されている参照テーブルにて予め記憶されているものであります。暗号化（又は復号化）を行う際には、送信先データ（送信元）をインデックスとして参照テーブルを検索し、所定の暗号化（復号化）プログラムを自動的に読み出して自動的に暗号化（復号化）を行うものであります。

このことは、送信者と受信者との順序対の関係に基づき、送信先データと暗号化プログラム、そして送信元データと復号化プログラムとをそれぞれ参照テーブルに対応付けして記憶させることにより初めて可能となるものであり、そのような記載は引用例には記載も示唆もされておりません。

本願発明においては、例えば、同じ２人の間の送受信であっても、送信者と受信者の関係が入れ替われば、異なる暗号化プログラム並びに複合化プログラムを用いることができるものであります。そして、引用例Ｄ1のように鍵復元窓口（recovery agent）を介在させることなく、直接送信者と受信者との間にて各個人が信頼する暗号化プログラムを使用できるようにしたものである。

審査官は、引用例Ｄ１に本願発明に記載の送信先データが既に開示されていると主張しておりますが、Ｄ１のテーブルにおける送信先データとは単純に送信先となる人が住んでいる国をあらわすものに過ぎません。本願発明における『送信先データ』とは、本願明細書中に記されているように、電子メールアドレスやＩＰアドレス等、そのデータにて送信されるファイルの最終的な送信先（受信者）が特定できる情報を意味するものである。

また、Ｄ１のテーブルにおいては、暗号化アルゴリズムの欄はあるが、復号化アルゴリズムの欄がない。本願発明においては、送信先データと暗号化プログラムとが、そして送信元データと復号化プログラムとがそれぞれ対応付けられているものである。つまり、一つの送受信相手に対して、暗号化に関する欄と復号化に関する欄と２つ別々に存在する。

上記に加え、庁指令において審査官も認めているように、本発明では受信した暗号化ファイルの送信元データをインデックスとして参照テーブルを検索し、解読アルゴリズム（復合化プログラム）を自動的に特定し、情報処理装置のメモリから読み出す手段を有している点で引用例とはさらに異なります。

請求項２－５について
請求項１の従属請求項なので、請求項１と同じ反論。

請求項６について
請求項１の構成を電話機に関連付けたものなので、請求項１と同じ理由で拒絶されております。すなわち、請求項１と同じ反論。

請求項７について
　　請求項６の従属請求項なので、請求項６と同じ反論。

７．３　勝利！（２００６年６月１５日）

　待ちに待ったメールが、やっと届きました。

宇山　靖政様
欧州出願番号：01130128.0-2413
発明の名称　：秘密通信システム
前略、いつも大変お世話になっております。
本日は嬉しい報告があります。
上記のヨーロッパ特許出願が許可されたとの通知が
ヨーロッパ特許庁より発行されました。
その欧州特許庁からの通知、並びに現地代理人からの
レターの概略を報告書としてまとめましたので
ご査収下さい。
追って郵送にて正式に報告させていただきますが、
良い結果ですので、取り急ぎメールにて
ご報告させていただきます。
　　草々

　これで、ヨーロッパの特許庁が認めてくれる見通しが立ちました。
しかし、ヨーロッパでは、英語、ドイツ語、フランス語の３カ国語で書類を提出しなくてはなりません。さらに、フィンランドにも提出しますので、フィンランド語の書類も必要です。
　翻訳料は１単語５０円前後らしいです。２００万円～３００万円くらいは用意する必要があります。まいったな。

日本では、特許として認められなかったので、これで１勝１敗です。

７．４　審査官のアドバイス

　承認の前に、条件を書いた書類がヨーロッパ特許庁から来ました。手書きで修正要求が書いてありました。修正の趣旨は表現を明確にするということでした。

７．５　ヨーロッパでの経過

09/08/2002 Publication in section I.1 EP bulletin published on 25/09/2002 [2002/39]

20/01/2004 New entry: Renewal fee paid
02/04/2004 Change - classification published on 19/05/2004 [2004/21]
09/04/2004 Publication of search report published on 26/05/2004 [2004/22]
16/07/2004 Request for examination filed published on 01/09/2004 [2004/36]
27/07/2004 New entry: Dispatch of examination report + time limit
10/12/2004 New entry: Dispatch of examination report + time limit
31/12/2004 Definitive list of designations published on 16/02/2005 [2005/07]

14/01/2005 New entry: Renewal fee paid
12/02/2005 New entry: Reply to examination report
04/03/2005 Deletion: Reply to examination report

04/03/2005 New entry: Dispatch of examination report + time limit
27/04/2005 Deletion: Fee payment for an extension state
27/04/2005 Deletion: Withdrawal of an extension state
23/07/2005 Change: Dispatch of examination report + time limit
23/09/2005 New entry: Reply to examination report

13/01/2006 New entry: Renewal fee paid
19/05/2006 Change - German title published on 21/06/2006 [2006/25]
23/05/2006 New entry: Dispatch of communication R.51(4) after July 1st, 2002
18/10/2006 New entry: Payment of fee for grant R.51

７．６　取得の表示（２００７年１月１１日）

EU 特許庁のＨＰにおいて、２００７年１月１１日付で特許認定されたとの表示がされました。文書の配布は２００７年２月１４日の予定だそうです。
始めてから７年。長い旅でした。

７．７　バレンタインデー（２００７年２月１４日）

２月１４日はバレンタインデーです。ヨーロッパでは、男女問わず贈り物をする習慣の国もあるのだそうです。ヨーロッパ特許庁は私に贈り物をくれました。

European patent granted: 14/02/2007

特許公報をバレンタインデーに発行してくれました。これで、正式に私が特許権者として認定された事になります。
最近のとっきょ公報はＣＤになっているようです。
私は、紙に印刷したものが、各国の特許庁に郵送されるのかなと思っていました。ＣＤはどんなところに配布されるのかは不明です。発明した本人には記念に１枚くらいくれないのかな？
ＣＤを額に入れて飾ろうと思っています。以前テレビで見たときは、特許認定証のようなものが額に入れて飾ってありました。それも欲しいのです。
教えてグーで聞いたら、登録証書が送られてくるとのことでしたが、楽しみです。

それにしても、アメリカ特許庁はのんびりしていますね。私の特許の有効期限が切れた頃には特許認定をしてもらえそうな気がしています。さすがはアメリカです。国益に反するものは許さないのでしょうか？

７．８　イギリス特許庁（２００７年３月１日）

７．９　特許証書到着（２００７年３月８日）

ついに、黄色い紙が届きました。そこには２月１４日は着付けで特許を認定すると書かれていました。
特許番号は
European Patent EP1244267
です。

７．１０　フィンランド特許庁（２００７年５月２４日）

フィンランド語への翻訳が完成し、フィンランド特許庁へ提出したと言う連絡が来ました。もちろん専門の翻訳家に依頼して作成した正式文書です。フィンランド特許庁から公開される文書の元です。そして、
翻訳料は、８８２１４９円です。
この金額は、ノキアの携帯電話がこの方式の暗号化を採用してくれるなら取り返せる金額でしょうが、難しいのかな？

７．１１　ドイツ特許庁（２００７年５月３１日）

ドイツ語への翻訳が完成し、ドイツ特許庁へ提出したと言う連絡が来ました。もちろん専門の翻訳家に依頼

して作成した正式文書です。ドイツ特許庁から公開される文書の元です。そして、
翻訳料は、３８７４６０円です。
この金額は、ノキアの携帯電話がこの方式の暗号化を採用してくれるなら取り返せる金額でしょうが、難しいのかな？

８．アメリカ決戦

８．１　最終反論（原案）（２００６年７月３１日）

アメリカ特許庁へ最終反論を書きました。これでだめなら諦めます。

アメリカ特許庁への最終反論（２００６年７月３１日）
Final amendment of the patent application to the US Patent Office. 31/7/2006_

宇山靖政
Uyama Yasumasa

最初に、拡張されたアドレス帳の性質、すなわち、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の項目があり、暗号化部分と復号化部分を独立した形で登録できます。このようなテーブル構造が、暗号通信における様々な問題を解決してくれます。

指摘された文献では、暗号ソフトのデータベースを使った対称的な暗号通信を考えていることが多い。このデータベースと、通信の対称性が様々な問題を引き起こします。なぜ問題が起こるかと言うことと、非対称的な暗号通信と拡張されたアドレス帳がどのようにしてこれらの問題を解決してくれるかを明らかにします。

The extended address book have terms of address, enciphering program, enciphering key, deciphering program and deciphering key. It is possible to register encryption part and decryption part independently. As the result of the structure of table, we can get settlements on many problems.

In old paper the enciphering communication is treated as symmetrical communication on the database of cipher programs.

This database and symmetrical structure is the cause of many problems.

We must consider non-symmetrical enciphering communication and extended address book. The non-symmetrical structure and extended address book gives settlements for many problems. I will show the reason in the next comments.

例１.

会社には顧客の個人情報など外部には出せない重要な情報があります。顧客からの情報の重要度が高ければ顧客からの情報は暗号化して会社に届きます。管理職が顧客からの暗号メールをすべて受け取って処理することは現実的ではありません。社員が顧客からの暗号化されたメールを受け取って処理してから、必要な点を管理職に報告するようにするのが現実的です。

社内の顧客情報を社員が社外に持ち出すのを防ぐには、社員が暗号化された情報を社外に送信することを禁止して、社外に暗号メールを送信できるのは管理職だけにする場合が考えられます。

この場合、会社員のメーラーは、受信した暗号メールを復号化できなくてはならないが、送信する情報を暗号化しないように設定できなくてはなりません。

暗号化部分と復号化部分を分離して、独立的に扱えばよいのです。

社員に暗号メールを送信するお客様のメーラーには、社員からの暗号メールは送られてこないのですから、お客のアドレス帳の、社員の行には、暗号化ソフトと暗号化鍵は登録されているが、復号化ソフトと復号化鍵の欄は空欄になっている。

社員のアドレス帳の、このお客に関する行は、暗号化ソフトと暗号化鍵の欄は空欄であるが、復号化ソフトと復号化鍵は登録されています。

Example 1:

The companies have important information, as personal information of the customers.

The outflow of each information must be prevented. In the case the customer send important information to the company, it must be enciphered.

The person, who is in the managerial post, can not accept all of these enciphered data. An employee will do the work instead of officer.

When the stuff received enciphered data from the customer, he deciphered and handled it. He will report

the result to his boss.
To prevent the outflow of each important information, forbid the sending enciphered e-mail to his employee. Only the person in the managerial post can send enciphered data.
In this case the employee's table contains deciphering program and deciphering key, but it is not contain the enciphering program and enciphering key.
The independence of encryption part and decryption part, and the existence of items address, enciphering program, enciphering key, deciphering program and deciphering key make the possibility of this.
On the side of customer, in the row of the employee in the table, enciphering program, enciphering key are exist, deciphering program and deciphering key are not exist.

例２.

暗号化された通信網（電話など）の場合を考えます。使用できる電話機やコンピュータの機能は短期間のうちに急激に向上します。

新しい機械は強力な計算機能を持っています。しかし、新しい電話機や、新しいコンピュータが発売されたとしても、それをすぐに購入できるのは暗号通信の利用者の一部だけです。利用者の中には１０年近く同じ機械を使い続ける人もいます。

計算能力の劣る電話機やコンピュータで利用できるのは複雑な計算を必要としない簡単な暗号化です。

A さんと、B さんと、C さんが暗号通信網のメンバーだとします。A さんと B さんは最新の機械を持っているが、C さんは古い機械を使っているとします。
A さんは、B さん、C さんと通信をするが、B さんは新しい機械を持っている人としか通信を行わないとします。

A さんと C さんが暗号通信をスムーズに行なうには、C さんの持っている機械の計算能力に合った簡単な暗号方式にしないと通信がスムーズにできません。なぜなら、強力な暗号化を選択すると、復号化に時間がかかりすぎるからです。C さんの機械では暗号化された信号を受け取ってから音声になって聞こえてくるまで時間がかかりすぎるので、電話としては機能しなくなります。

暗号データベースは、最新の機械を持っている人たちのうちで、古い機械を持っている人との通信を行なわない人たちにとって無駄の多いものとなります。使わない古いソフトを保存することがメモリーの無駄使いなのです。

もし、C さんを除いて最新式の機械を持っている人だけで暗号通信を行なうならば、メモリーが有効に活用できるので、もっと強力な暗号化を使用できます。しかし、C さんを除外するのでは新しい機械が出るたびに通信網が破壊されてしまいます。

全員が共通の暗号化ソフトを利用する方式をあきらめて、暗号ソフトはそれぞれの利用者が独自に管理するべきです。A さんは A に合ったものを、C さんは C さんに合ったものを利用するようにすべきです。

従って、できるだけ強力な暗号通信を実現するには、高機能の機械を持っている人とはその計算能力を十分に活用するような強力な暗号で通信し、自分が高機能な機械を持っていても、古い機械を持っている人とは、簡単な計算機能でも利用できる簡単な暗号を利用しなくてはなりません。

したがって、A さんのアドレス帳は次のようになります。
B さんの行の暗号ソフトは強力なものであり、暗号化鍵は長いものとなります。もちろんB さんの行の復号化ソフトは強力なものであり、復号化鍵は長いものとなります。もちろん、暗号化ソフトと復号化ソフトは、別の暗号アルゴリズムを使用したソフトでかまいません。
C さんの行の暗号ソフトは簡単なもので、暗号化鍵は短いものが登録されます。もちろんC さんの行の復号化ソフトは簡単なものであり、復号化鍵は短いものとなります。もちろん、暗号化ソフトと復号化ソフトは、別の暗号アルゴリズムを使用したソフトでかまいません。

C さんは、自分の古い機械に合わせてすべて簡単な暗号ソフトだけを自分の機械に登録しなくてはなりません。

したがって、C さんのアドレス帳は次のようになります。
B さんとC さんは通信を行わないので、B さんの行は、存在しません。
A さんの行の暗号ソフトは簡単なもので、暗号化鍵は短いものが登録されます。もちろんA さんの行の復号化ソフトは簡単なものであり、復号化鍵は短いものとなります。もちろん、暗号化ソフトと復号化ソフトは、別の暗号アルゴリズムを使用したソフトでかまいません。

Ｂさんの通信相手が強力な最新の機械を持っている人だけなので、Ｂさんの機械に登録される暗号ソフトはすべて強力なものにできます。使うことの無い古いソフトを自分のコンピュータに置く必要はありません。

したがて、拡張されたアドレス帳は、A さん、Ｂさん、Ｃさんすべて異なり、共通のデータベースとは無縁であることは明らかです。

暗号ソフトのデータベースをつくり、暗号ソフトを全体的に統一する発想は、データベースの更新が困難な場合は全体が古い機械の計算能力によって時代遅れのままにされてしまうか、または更新と共に、古い暗号ソフトを廃棄するなら古い機械を持っている人を暗号通信網から追放するかのどちらかの方法を選ばなくてはなりません。追加しながらも廃棄しないなら不要な暗号ソフトを大量に保管することになり、メモリーを無駄に使用することになります。

もちろんデータベースの更新､統一がもっとも困難な課題であることは例３から明らかとなります。しかし、暗号データベースによる統一を捨てて、私が提案したところの、拡張されたアドレス帳を利用すれば、そこには暗号化欄と復号化欄が通信相手ごとに自由に設定できる仕組みが存在するので、高機能の機械を持っている人に対しては強力な暗号ソフトを登録し、古い機械を持っている人に対しては簡単な暗号ソフトを設定できるので、新しいものも古いものもその機能を十分に生かしながら、暗号通信網の中で共存できます。

Example 2:

In the case of cipher communication (telephone machine and e-mail etc.). The improvement of the machine is progressing day by day.

New machine is powerful, but only few persons can purchase it, as soon as it appears.
Poor person uses old machine for a long time.

Old machine can treat simple cipher program. It is not appropriate to use powerful encryption algorithm. If new powerful program is used on the old machine, too long time is needed for encryption and decryption.

We consider the case, three persons A, B and C are the members of cipher communication network. The person A and B have new machine and C has old machine.

Suppose that, A send encrypted data to B and C. All of B's friends having new machine.
B does not send data to C.

For the communication of A and C, C can not use powerful cipher programs. If A want to use new cipher program to the communication with C, it is impossible for C's machine can not decrypt it in an appropriate time.

In the case, using database of crypt program, the participation of C is cause of futile. All most of the member of this crypt network having new machines, the preservation of old program is waste of memory.

The exclusion of C gives efficient network. This is destruction of network. The database of crypt programs is the reason of this destruction.

We should repeal the database of crypt program. My extended address book will be able to give the system, which does not use the database of crypt programs.

Each person should control his own address book. This gives us most efficient cipher network. The member possesses least and powerful crypt programs. Who has powerful machine, and all his friends has powerful machine, he can use powerful programs only.

In A's address book, the row of C, contains simple enciphering program, short enciphering key, simple deciphering program and short deciphering key. In A's address book, the row of B, contains powerful enciphering program, long enciphering key, powerful deciphering program and long deciphering key.

In C's address book, the row of A, have simple enciphering program, and simple deciphering program. The row of B does not exist in C's address book

In B's address book, all of the programs are powerful and new.

As the result the table of C is not copy of the table of A. The address book A, B and C are different.

The idea of database of crypt programs is the reason of many troubles. Only the extended address book gives us the real construction of cipher network. And give us the harmony of old machine and new machine.

例３.
　暗号データベースには、さらに困難な問題があります。１つ目は、暗号の使用に関する制限です。
　国によって暗号通信に関する政策には大きな違いがあります。自由に暗号化された通信を行なえる国や、暗号化されたデータを受信するのは可能だが送信することが許されない国や、その逆の国。暗号化されたデータの送信も受信も許されない国や、暗号の使用は許可されるが、アルゴリズムや暗号化鍵、復号化鍵の扱いに制限が付く国などいろいろです。
　国際化が叫ばれる時代です。通信が国境を越えるのは普通のことです。暗号ソフトの共通化は外国の友人を、犯罪者にしかねません。
　たとえば、鍵の長さに制限が付く国に住んでいる人に対しては、その制限内の長さの暗号化鍵を使い、特に制限の無い国に住んでいる友人とは、機械の能力の許す限り強力な暗号を作り出すための長い鍵を使って暗号通信をすべきです。
　暗号化鍵と復号化鍵の提出を求める国にすんでいる人が、この暗号通信網に参加した場合は、もっとひどいことになります。集積された鍵の情報が流出すると、暗号データベースに登録されている暗号をすべて復号化できる立場の人が存在することになるからです。
　この点から見ても、全体を統一してはいけません。各国の法律を守りながらも、できる限り安全な通信を実現するには、各人が自分のアドレス帳を自由に管理し、最適な暗号ソフトの登録を行なえる必要があります。しかも、自分と直接暗号通信をする人の分だけを管理するのです。もしも、ある人のコンピュータがクラッキングされても、全体の暗号通信網に対する影響は少ない。クラッキングされた人と直接通信を行なう人々が、クラッキングされた人の行だけをあたらいいソフト、新しい鍵に置き換えればよいのです。
　私の方法ならば、鍵の提出による影響もわずかです。規制の無い国に住むＡさんの通信相手の一人Ｂさんが鍵の提出が義務付けられている国に住んでいるなら、Ｂさんとの通信に使う鍵だけはＢさんを通じてＢさんの国に提出しますので、Ｂさんの分だけは影響を受けますが、他の部分、提出を要求しない国に住んでいる人との通信に使う部分の秘密を保てるからです。
　情報を集積しないからこそ、安全性が強固となるのです。新しいコンピュータを入手した人は、すぐに更新できます。すくなくとも鍵の長さを長くすることは可能です。

　２番目は、輸出の問題点です。
　さらに、暗号ソフトを輸出する事に関してはさらに強い制限が付きます。暗号データベースが共通であるときは、その中に輸出禁止となる暗号ソフトが含まれます。誰がデータベースのコピーを配布するのでしょうか？
　フリーソフトとして、インターネットからダウンロードする形で配布しようとしても、強力な暗号ソフトを外国に輸出する行為と考えられて暗号ソフトの配布者が犯罪者になってしまう国もあります
　この暗号データベースを使った暗号通信網を構築したとたんに、犯罪者となってしまいます。輸出が許可されるような簡単なものだけを暗号データベースから抜き出して配布すると、共通性が無くなり、暗号データベースとして機能しなくなります。暗号データベースは、国際的な暗号通信網を構築できません。
　拡張されたアドレス帳を使う方法では、この問題点に対して次のように対応します。法的な規制の無い国と法律の規制の無い国の間では強固な暗号を使えます。規制のある場合は、規制内でもっとも強力なものを選択すればよい。暗号データを受け取る場合の規制は弱いことが多いでしょう。なぜなら、暗号化されたスパムメールで沢山の人が犯罪者にされてしまうから。送信すると時の規制と、受信するときの規制に違いがあるならそれに合った形でもっとも強固なものを選択できます。

３番目は著作権の問題です。
　暗号ソフトの共有は、暗号ソフトの製作者の著作権を無視しかねません。コピーを配布するときの料金の支払いは誰がどのようにして負担するのでしょうか。
　また、この暗号データベースに新しい暗号を追加するときは、そのコピーを世界中に配布しなくてはなりません。著作権者にお金を支払うことを全員が納得するでしょうか？
支払いを拒否した人はこの暗号通信網から除外されてしまいます。
　全員がコピーを共有するのでは、指定されたもの以外の暗号ソフトの使用が不可能になります。標準とされる暗号ソフトは、DES から Triple-DES、さらに AES のように目まぐるしく変化しています。RSA 暗号でも、安全とされる鍵の長さは、５１２ビット、１０２４ビット、２０４８ビットと、どんどん長くなっています。
　古くなった暗号方式を捨てて、新しい暗号方式を採用するとき、それを違法コピーするか、全員が購入するかのどちらかになりかねません。あるいはフリーソフトとして、インターネットからダウンロードする形で配

布しようとしても、強力な暗号ソフトを外国に輸出する行為と考えられて暗号ソフトの配布者が犯罪者になってしまう国もあります。郵送で国内のみ配布可能と言うような国もあります。郵送で新しいソフトを配布するだけでも費用がかかります。郵送料は誰が負担するのでしょうか。

新しい強力な暗号ソフトを利用するには、開発者の適正な利益を保証できる方法が必要です。私が提案している拡張されたアドレス帳とそれにあった暗号ソフトならばこれが可能となります。

新しい暗号が開発されたら、必要な人はお金を払ってすぐに購入できます。それをすぐに利用できて、しかも開発者の利益を保証するような仕組みを考えると、暗号ソフトの分離が必要となります。暗号化部分と、復号化部分の分離です。そして復号化部分の再配布禁止、暗号化部分の再配布は自由。鍵の作成と配布が可能でなくてはなりません。

この分離された暗号ソフトに対応するには、暗号化ソフトの欄、暗号化鍵の欄、復号化ソフトの欄、復号化鍵の欄が独立していることが必要なのです。

さらに、長い鍵の RAS は強力な計算機能を必要とします。古い機械では対応できません。鍵まで含めての暗号ソフト共有を優先すると、ここでも通信網内に存在する古い機械が、暗号通信の安全性を脅かします。

Example 3:

The database of crypt programs is the account of more difficult problems.

The first problem is the restriction of law. The legislation in each nation is distinct very much.

In a country, it is possible to send ciphered data without any restriction. Another country allows accept encrypted data, but forbid send an encrypted data.

There may be the country having contrary restriction.

A country forbids sending and receiving ciphered data.

A country allows accepting ciphered data and sending ciphered data, with the condition of submission of encryption key and decryption key.

The internationalization of communication is normal. The shearing of common database of crypt programs may make a criminal.

In the case of a country, there is restriction of the length of key, appropriate length key must be used in the communication.

In the case of a country, the exhibition of key is duty, who is living in the country should exhibit the keys, which he has.

If the database is used, the outflow all of the keys may occur. This is a weak point of database of crypt programs.

In this point of view, the unification of database of crypt programs must be discarded.

Each person must use crypt programs within the law. The total safety of the crypt communication network must be protected.

Each person should keep least programs and keys, which is needed, in his computer. It must be possible to manage his extended address book. This condition gives us the safety of the total cipher network.

In this case, if a person's computer cracked, the repair is easy. Only the person and his friends rewrite their address book, and change the crypt programs.

By my method the influence of submission of keys become minimum. If A is living a country where he can use cipher free. And B is living a country, where he must submit the key. A's key is submitted from B to the government which is used for the communication with B.

A can use powerful cipher program for the communication with other person, that algorithm and key is different from the algorithm and key which is used in the communication with B.

As the result, the influence of the submitting of key becomes least. It is the reason of the safety of the cipher network, that which does not have accumulated cipher programs.

In my system, if a person get a new machine, who can change his cipher program to more powerful program. This behavior produces more powerful cipher network.

The second problem is export.

There is more strong restriction on the export of crypt programs. In the case, the network uses the common database of cipher programs. Who can distribute the database of cipher programs, without guilty decision of judge?

The upload of cipher programs to the Internet, become to crime. No one can construct crypt network, without guilty decision of judge.

Very weak crypt programs may be able to export. But the total database of cipher programs must exist in each person's computer.

For this reason, the database of cipher program can not construct cipher network on the Internet.

Extended address book can solve this problem. Each person can select the cipher programs for registration into his address book. Considering the conditions of his friends.

The third problem is copyright and patent right.
In the case, every one has the common database, how much the price of database?
The database will include many programs, which will not be used. The person must pay the price of the software, which is never used. The new participant will not use the old software, because he will have new machine.
In the case of adding new crypt program to this database, dose every one agree to this?
If this program is new and need powerful CPU, the person who has old computer, will not agree to add this program. He must pay the cost of software, which he never uses.
In this cases, the person who is having old computer, should be excluded or not. This is the reason of breakdown of cipher network.
The database must be unique and all members must have it. When adding new program, the distribution of the copy is difficult. If it is powerful, we can not distribute it using Internet. In the case, using post for distribution of program, who should pay the cost of the post.
The programs are progressing from DES, Triple-DES to AES and RSA to ECC. We should not use the database of crypt programs.
For the development of new crypt program, much more time and much more cost. The user must pay the cost. This is the foundation of new and powerful crypt program.
My extended address book and separated crypt program make it possible. Every user can buy new program and distributes encryption part and key to his friend, and register the decryption part and key.
It is clear safety RSA need long key too old computer can not correspond to this long key. An old computer in the net work, using database, is the cause of weakness of this net work.

各文献の問題点
１．Vidrascu
col. 6 line21-31
When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.
The transmission is next sent to the interface 31;
When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30.
user@domein 形式のメールが扱えるようになったのは１９８６年です。ＩＰアドレスは@の後ろのdomein に対応します。したがって、
user11@domein1 と user12@domein1 は同じＩＰアドレスを持ったメールサーバーを使い、
user21@domein2 と user22@domein2 の使うメールサーバーも同じＩＰアドレスを持つことになります。
したがって、
user11@domein1 から user21@domein2 への通信と
user12@domein1 から user22@domein2 への通信では同じ暗号化が行なわれます。
同じプロバイダーを利用する人はとても多く、同じＩＰアドレスを使うユーザーが必然的に多くなります。よって、同じ方式、同じ暗号化鍵で大量のデータを暗号化して送信することになります。こらは暗号が解読されてしまう大きな原因になります。
ＩＰアドレスとＩＰアドレスの間の対応はメールサーバーとメールサーバーの間の対応を意味しています。これでは暗号ソフトを変更できるのはメールサーバーの管理者だけになってしまいます。同じプロバイダーのメールサーバーを利用している人たちはたくさんいます。そのうちの２つのメールサーバーを利用している人たちのグループは同じ暗号化ソフトを使うことになります。これでは暗号ソフトを、電子メールの各ユーザーが自由に変更することは出来ません。解読技術が見つかったらすぐに変更できなくてはなりません。
電子メールのユーザーが自分にあった暗号ソフトを選んで自由に利用するためには、ＩＰアドレスとＩＰ

アドレスの対応ではなく、メールアドレスとメールアドレスの間の対応に従って暗号化ソフトや暗号化鍵を設定できなくてはなりません。自分が新しい機械を購入したらより強固な暗号に変更出来るようにしておくべきです。これが出来なければ法律的な制限が異なる外国に住んでいる人との暗号通信がトラブルの元になってしまいます。自分のアドレス帳に登録されている様々な国の人々と円滑な暗号通信を実現するには電子メールの利用者が自分のアドレス帳の中で自由に、きめ細かく設定できなくてはなりません。

さらに、暗号の鍵をＩＰアドレスに合わせて選ぶことは書かれているが、暗号ソフトをＩＰアドレスにあわせて選択するとは書かれていない。この中で現れる暗号化ソフトはＤＥＳだけであり、同じ相手似たいする送信時と、同じ相手からの受信時で異なるアルゴリズムの暗号を使うような記述は存在しない。

この方式は更なる問題点を含んでいます。安全なネットワークと安全でないネットワークを分けて考えて、その境界で暗号化と復号化を行なおうとしている点です。問題は、安全とされるローカルなネットワークが存在するか否かです。会社の顧客情報は高い値段で売り買いされます。社員が安全とされるローカルなネットワーク内で暗号化されないで流れているパケットを盗聴し外部に販売する可能性も十分に考えられます。情報漏洩のほとんどは社内の人間が機密情報を社外に流出させるような形です。したがって、安全なネットワークと安全でないネットワークの境界など存在しません。自分のパソコンから出る時には暗号化されている必要があるのです。従って、暗号ソフトの管理は各パソコンで直接しなくてはならないし、各人が自分のパソコンの中に拡張されたアドレス帳を持っている必要があるのです。

col.12　line 1-11 では
if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.
verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and
if the first and second keys exist, deciphering a received message with the second key and otherwise reject a received message.

すぐに問題となるのは、第１の鍵と第２の鍵がどこに存在することを誰がどの様にして確認できるかということである。

第１の鍵は、送信時の暗号化に使われ、第２の鍵は暗号化されたデータを受け取る側での復号化に使われます。Ａさんがｂさんに暗号化したデータを送るとします。

Ａさんが自分の所に第１の鍵と第２の鍵があることを確認しても、第２の鍵がＢさんの所にあることを確認したことにはなりません。Ａ さんが送信前にＢさんに電話をして確認しても、Ｂさんが本当に確認しているか、嘘をついているかの判断は出来ません。Ａさんは、送信前にＢさんの所に出かけて、第２の鍵の存在を自分で確認しなくては成りません。
国際的な通信では実用になりません。

そもそも、確認不可能な第２の鍵を調べる必要はありません。第１の鍵があったらそれで暗号化して送れば十分です。上手く届かなければＢさんから知らせがあります。

さらに、ここで確認しているのは、鍵の存在だけで暗号ソフトの存在を確認してはいない。もし暗号ソフトを何種類も使ったり、暗号化する場合と暗号化しない場合があるのなら、必ず確認しなくてはならない項目である。その記述が無いのは、暗号ソフトはＤＥＳのみを考えていて、暗号ソフトの種類を、送信のとき（たとえばＤＥＳ）と受信のとき（たとえばＡＥＳ）で変更しようと言うことは、まったく考えていないからです。暗号ソフトの変更を考えるなら、誰がどのような立場で決定し、どのようにして暗号ソフトの変更や登録をするかについての記述が必要です。

鍵を考えるなら、４つ考えなくてはなりません。
１．Ａさんがｂさんへ送るデータを暗号化する時に使う鍵。
２．Ｂさんが Ａさんから受け取ったデータを復号化するときに使う鍵
３．Ｂさんが Ａさんに送るデータを暗号化するときに使う鍵
４．Ａさんがｂさんから受け取ったデータを復号化するときに使う鍵。

暗号ソフトも同様です。
１．Ａさんがｂさんへ送るデータを暗号化する時に使うソフト。
２．Ｂさんが Ａさんから受け取ったデータを復号化するときに使うソフト。
３．Ｂさんが Ａさんに送るデータを暗号化するときに使うソフト。
４．Ａさんがｂさんから受け取ったデータを復号化するときに使うソフト。

私の方法では、遠隔地にあるものの存在を確認することはしません。過去にあったとしてもいま存在することの証明にはならないからです。
送信の時に確認するのは、暗号化鍵と暗号化ソフトです。暗号データを受け取っても、暗号化されないデータのみを返してくる人もいるのだから。
受信時には、復号化鍵と復号化ソフトの存在だけを考えるのが当然である。その人から、暗号化されたデータを受け取れても、その人に対して暗号化されたデータを送信できない場合もあるからです。暗号化されたデータを受け取っただけで犯罪者扱いされてしまう国の存在も考えられます。
暗号メールは受け取れないが、暗号メールを送信できる場合も考えられます。したがって、送信のときは暗号化鍵と暗号化ソフトだけを、受信のときは復号化鍵と復号化ソフトだけを調べるようにしないと、国境を越えた暗号通信（たとえ片方だけが暗号化可能だとしても）が出来なくなります。
この文献では、公開鍵方式の暗号プログラム使うことにしたとき、大きな問題が生じます。公開鍵方式では第２の鍵は秘密鍵で、それを送信者の手もとにおくことはしません。この方法は公開鍵方式に対応できません。公開鍵方式に対応できないものはもはや時代遅れです。

２．Schier
col.9 line67　col.10 line1-7 では
The receiving device 78 includes a copy of table 80. The copy of table 80 is then used at step 132 to retrieve the first indicated encryption algorithm from the encryption algorithm column 84. This encryption algorithm is then loaded into the encryption decryption engine 30 and is executed by central processing unit 20 to encrypt outgoing communication and decrypt incoming communication at step 134.
この問題点は、テーブル 80 のコピーを持っていることである。テーブルに記載された暗号化アルゴリズムの番号によって指定される暗号化ソフトと同じ番号で指定される復号化ソフトが 78 と 72 の両方に存在することである。
このことは、ＡさんとＢさんが同じテーブルを持つなら、ＢさんとＣさんも同じテーブルをもち、。。。となって、全員が同じテーブルを持つことになります。
暗号通信網に参加する人全員が、共通のテーブルを持つのは幾つかの点で問題を含みます。

１．このテーブルは追加、削除、更新ができるのか？
２．追加、削除、更新の決定は誰がするのか？
３．追加したソフトはどのようにして配布するのか？
４．著作権料はだれが、どのように払うのか？
５．国を超えた配布は犯罪行為にならないか？
６．テーブルが新しい強力な暗号に変わったとき、古い機械の持ち主は利用できるか？
７．新しい機械を持っている人は、使えない古い暗号を保存する必要がなぜあるのか？

テーブルを共有するとこのような問題を突きつけられてしまいます。テーブルの共有をあきらめなくてはなりません。逆に、決して共有しないことが、強固な暗号通信網を構築する基本原理なのです。
各人が拡張されたアドレス帳を自分の分だけ管理することが、上記の矛盾を解決してくれます。

コピーを使うということは、この暗号通信網に参加するすべての機械が同じテーブルと同じ暗号化ソフト、同じ復号化ソフトを共有することになる。
この暗号化通信網が、旧式の機械の参加を認めるからには低い計算能力でも実用となる弱い暗号化を使うことになってしまう。
すべての機械がコピーを持つからには、その暗号化通信網に参加する機械の台数が多くなると、新しい暗号アルゴリズムに変更することが極めて困難になる。なぜなら、暗号化ソフトや復号化ソフトの著作権料を払える人と払えない人がいるからであり、また異なる国の法律はそのアルゴリズムの使用を許す場合と、許さない場合がある。強力な暗号ソフトを国外に持ち出すことを禁止している国もあるし、自由に持ち出すことを許している国もある。コピーをインターネットで配布すると犯罪行為になってしまう事も多い。
ある国では、受信するデータの暗号化については問題にしないが、送信するデータの暗号化には制限をかける場合も考えられます。
そうすると、Ａさんが住んでいる国では自由に暗号が使え、Ｂさんが住んでいる国では受信は自由だが送信するときは簡単な暗号化しか許されない場合もあります。このとき
Ａさんのアドレス帳の暗号化ソフトは最新の強力なものを登録し、暗号化鍵も長いものを登録できます。しかし、Ｂさんのところの復号化ソフトは簡単なものでなくてはならないし、また復号化鍵の長さは、短いもので

なくてはなりません。
Aさんの暗号化のテーブルと暗号ソフトをBさんに送ってしまったら、Bさんがそれを使っただけで犯罪者になってしまいます。暗号に関して、法律的な制限が異なるのが現実です。この中で少しでも有効な暗号通信網をつくるには、各人が独自に管理する拡張されたアドレス帳（電話番号帳）が必要なのです。
さらに、復号化鍵を国の公共機関へ提出することを求めている国もある。暗号データベースのコピーを使うならば、鍵を提出させた国はすべての通信を解読することができる。このような集積された復号化鍵、暗号化鍵の情報は高い値段で販売できるので、公共機関内部からの情報漏洩の心配も起こる。また、ネットワークに参加する人全員が暗号化と復号化を共有するなら全員が解読する能力をもつことになってしまう。
以上の理由で、この方法は国際化された通信には向かない。コピーをあきらめなくてはならない。しかし、私が拡張されたアドレス帳の考えを使えばこの困難は解決できる。
単に電話番号から暗号化ソフトを呼び出すだけなら簡単な話であるが、暗号ソフトを暗号化部分と復号化部分に分離し、それに合う形でアドレス帳を拡張し暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵を登録できるようにしたことがもっとも重要な点であると確信しています。この点についてはまったく記述されていない。コピーを配るような記述は自ら暗号通信網を破壊するようなものであり、決して行なってはならない。

私の方法では、
AさんとBさんが新しい電話機を買って、Cさんが古い電話機を使いつづけているとします。このとき、私の提案する方式では、Aさんの持っているアドレス帳のBさんの行に登録される暗号ソフトと暗号化鍵は最新のものを登録してそれを使うことが出来ます。
Cさんは古い電話機しか持っていないので、Aさんと暗号通信を行なうときもBさんと暗号通信を行なうときも簡単な暗号化方式を使わないます。古い電話機では復号化や暗号化に時間がかかりすぎてしまうので、新しい強固な暗号を使うことはできません。したがって、Cさんの拡張されたアドレス帳のBさんの行には古い簡単な暗号ソフトが登録されていなくてはなりません。
したがって、Aさんの拡張されたアドレス帳とCさんの拡張されたアドレス帳は待ったくべつのものでありコピーではありません。
各個人が自分の持っている電話機の性能と自分の通信相手の持っている電話機の性能に合わせてそれぞれ暗号化、復号化の設定をできるようにしておくほうが、暗号通信網全体としての強度は向上します。
この点から見ても、アドレス帳のあるいは電話番号帳の項目に、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の５つの項目が必要となるのです。

各人の暗号表は各人で管理する。共通化はしない。これが大切である。こうすれば新しい機械がすぐに力を発揮できる環境を作れます。古い機械が暗号通信網の安全性を脅かしてはいけない。各人が自分と自分の通信相手の状況にあった形で拡張されたアドレス帳の内容を調整できることが大切です。
この点から見ても、拡張されたアドレス帳は暗号通信の安全性を飛躍的に改善し、しかも国際暗号通信での法律的な問題点も解決できる方法であり、従来の技術よりも優れたものであると確信しています。
これが可能となったのは、AさんとBさんの会話を、AさんからBさんへの方向性を持った情報伝達と、BさんからAさんへの方向性を持った情報伝達の２つに分けてそれぞれ別々の暗号化アルゴリズムで暗号化しようと考えたからです。
記号化すれば、AとBの会話｛A、B｝を（A→B）と（B→A）の２つに分解して、それぞれに対して別々の暗号化を導入したということです。

３．Leppek
col.2 lines 19-23, 51-55, col.4 lines 14-17
The fundamental machine of the virtual encryption scheme of the involves the generation of a sequence of the access code, with immediately successive ones of the access code of the sequence being different from one another.

col.2 lines 51-55,
The order of the encryptors within the sequence to which the data is applied may very as desired, and the sequence may 'toggle' or switch back and forth between the same set of encryption operators as part of its overall encryption flow.

col.4 lines 14-17
The encryption routines 110, in and of themselves, need not be any particular type of encryption algorithm and may be conventional encryption operations, such as PGP, DES, etc. routines, as non-limiting

examples.

データが多重暗号化されて送信されてゆく場面と、受信者のところに到着してから、復号化されてゆく場面だけを考えればよく似た現象が起こります。

この文献の方法には沢山の問題点があり、現実的ではありません。
１．データベースは全員共通なのか？
共通ならば、著作権と違法コピーの問題はどのようにして解決するのか？
経済格差や法律の壁はどのようにして超えるのか？
共通でないなら、鍵から番号を作り出すアルゴリズムはどのようにしてこのアンバランスに対応するのか？

２．テーブルサイズに制限は無いのか？
同じアルゴリズムで暗号化鍵の長さを変更するときはどうするのか？
どんどん長くなる鍵ＲＳＡでは、現在 2048 ビット鍵だが近い将来８０００ビット程度になると思われる。
データベースの表の長さは２の 8000 乗になる可能性があり、全体を管理できなくなる。
２の８０００乗は約１０の 2400 乗であり、現在パソコンのハードディスクは１００Ｇバイト程度で、これは 100,000,000,000 バイトであり 10 の 11 乗にすぎない。
管理しきれないばかりでなくこのデータベースを表で管理するのは無理である。

２．新しいソフトの追加は可能なのか？
追加したことをどのように知らせるのか？
沢山の人が新しいものをそれぞれランダムに追加したとき登録の順序はどうなるか？
番号はどのようにしたら共通にできるか？
それとも共通の番号にしなくても良いのか？
使用する新しいソフトの番号はどのようにして決まるのか？
新しく、利用することにしたソフトはどのように配布するのか？

３．古いソフトの削除は可能なのか？
いつまで古いものを保存しなくてはならないのか？
削除するための全員の合意はとれるのか？
削除したら番号はつめるのかつめないのか？

４．書き換えは可能なのか？
かってに削除しその場所に新しく登録できるのか？
出来たときは番号の変更を誰にどのように知らせるのか？
利用するソフトはどのように再配布するのか？

５．変更後の番号の決定はどうなるのか？
番号を生成するアルゴリズムは同じ鍵に対して異なる列を生成してくれるのか？
削除したらそれを除外した新しい列を生成してくれるか？
この調整はどのようにしたら可能なのか？

５．番号を決定する SUPERVISOPY ENCRYPTION MANAGER のアルゴリズムはどうなっているのか？
データベースに新しい暗号を追加できるのか？
追加したとき、生成する列の変更、調整可能なアルゴリズムか？
存在するデータベース全体に同じものが追加できるのか？（法律、著作権）
新しい参加者にどのように対応するのか？
番号生成アルゴリズムは新しい参加者に対してどのように対応できるのか？
その新しい参加者は、古くて使い物にならないソフトを登録しなければならないのか？

６．ENCR.KEY は何を使うのか？
アドレスなのか？
新しい参加者に対してどのようにして１６０の列が生成されるのか？
新しい参加者に対して生成される番号はすでに決まっているのか？

それとも、参加者が自由に生成される列を設定できるのか？
設定できなければ、時代遅れの暗号を使うことを強制されるのか？

７．誰が使っているものも、同じキーを入れたら同じ列を生成するのだろうか？
盗聴者が持っているものも同じ様に、復号化の順番を教えてくれるのだろうか？
それとも、そのデータは、利用者が見ることは出来ないのだろうか？
操作することは出来ないのだろうか？
変更は出来ないのだろうか？

以上見てきたように、たとえ受信者のアドレスから暗号化する順番を決めたとしても、データベースの構造に問題点があり、更新、削除、新規登録などにどのように対応するかが明確ではなく。実用的ではない。

結論は、
１．全体を管理するデータベースは使用しない。
２．各人が自分の通信に使う分だけを管理するようにしなくてはならない。
３．各人が、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵を各通信相手ごとに管理する。

このように、拡張された自分のアドレス帳に基づいた私の方法が、登録、更新が簡単で、しかも違法コピーの問題、法律の問題をクリアーできる唯一の方法である。

４．Collins
Col.5 lines 43-63
The Internet e-mail gateway 240 converts the SMS message 100 to an Internet e-mail message having the format 150. To accomplish this, the Internet e-mail gateway 240 determines an Internet e-mail address for the cellular telephone 210 and for the Internet station 280. Regarding the Internet e-mail address for the cellular telephone 210, the Internet e-mail gateway 240 adds the domain name for the Internet e-mail gateway 240 to the SMS address for the cellular telephone located in the sender SMS address field 110 of the SMS message. To determine the Internet e-mail address for the Internet Station 280, the Internet e-mail gateway 240 uses a messaging service lookup table having a first column with a list of SMS addresses, where each SMS address corresponds to an Internet e-mail address in a second column of the lookup table. The Internet e-mail gateway 240 locates the SMS address of the Internet station 280 in the first column of the lookup table. The Internet e-mail address for the Internet station 280 is retrieved from the second column of the lookup table at a location corresponding to the Internet station SMS address of the first column.
ここでは、単にＳＭＳアドレスと、インターネットメールアドレスとの対応関係が記述されているだけです。

これでは、拡張されたアドレス帳の構造は発見できない。
なぜなら、ＡさんとＢさんの通信において、ＡさんがＢさんに呼びかけるときの信号伝達を暗号化する方法と、それに対応して、ＢさんがＡさんに答えるときの信号伝達を暗号化する方法を同じにすることも可能であり、そのほうが簡単なのになぜ、別々にしなくてはならないかが解明されていないからです。

この解明が不十分だから、先行技術は様々な問題点を含んでいて暗号通信の発展を困難なものにしてしまっているのです。暗号通信の発展を望みながらも、それを妨害してしまうような構造のものしか提供されていません。
その必要性に気づくのは困難であり。これこそが私の発明の核心的な部分である。著作権、国ごとの法律の違い、機械の計算能力の違い、などに対応しながらも全体として強固な暗号通信網を構築するには、
アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の５項目を備えた、拡張されたアドレス帳を各個人が自分の持っている機械の性能と、各個人それぞれの通信相手のことだけを考えて自由に高度化できる方式こそが必要であり、これによって様々な問題を解決できます。

この考えは、ＡさんとＢさんの通信に２つの異なる暗号化方式を用いると考えたからこそ、アドレス帳の５

項目の必要性となり、それによってソフトウエアの暗号化部分と復号化部分の分離につながりました。これによって著作権問題や不正コピー問題の解決手段を提供できたのです。この点は大きな進歩であると確信します。

このような拡張されたアドレス帳についての論点は、審査官が示された文献のなかには、まったく解明されていません。したがって、私の方法は特許されるのにふさわしい進歩性を持っていると思っています。

５．Keats
Each network element has its own SID (maintained for example in a memory element under control of the SID manager) and also maintains a database, table or other suitable structure in another memory element, such as a cache, which maps TIDs to IP address for IP connections from the particular network element to another network element in the network, as described in detail below.

テーブル構造を利用してデータを変換するのは、昔からある。住所録、電話帳、様々な数表などなど。
しかし、暗号通信における様々な問題点が何であり、それらを解決するには、どのような構造のテーブルが必要なのかについては何も述べられていない。
解決すべき問題点すらも気が付いていないといわなくてはならない。

拒絶理由への反論
１．特になし

２．文献の内容は拒絶理由に当たらない。なぜなら、進歩性が十分あると考えるからです。
I believe only my method can construct safety cipher network. The contents of literatures don't exhibit really usable system. They have much more confusion.

３．特になし

４．特になし

５．特になし

６．特になし

７．特になし

８．請求項の言葉の中にテーブルの構造が示されていないことは指摘の通りです。さらに趣旨を明確にするために記述を変更します。
According to your advice, I will add the sentence for explain the structure of the table.

クレーム１（最終的な修正案、宇山作）は、
1. A computer program stored on an information processing device, having a memory and means for communication over a network, characterized in that
the memory having a reference table storing plurality of kinds of enciphering and deciphering programs,

As Fig. 1,
the reference table must have next items (columns) of address, enciphering key, enciphering program, deciphering key, and deciphering program. The table may have other columns.
In this table each address can appear only one time in the column of address. When an address is given, if it is exist, we can decide a row uniquely in this table.
In the selected row, the enciphering program and deciphering program should be different kind of

algorithm's programs, as DES and RSA.
The relationship between tables.
Case user1 send enciphered data to user2, and user2 receive data from user1 and decipher it, and user2 send encrypted data to user1 and user1 receive it.
In user1' table, a row must have user2's address, in this row encryption program and encryption key must exist, the term of decryption program and decryption key must be exist.
In user2's table, a row must have user1's address, in this row encryption program and encryption key must exist, the decryption program and decryption key must exist.
The two programs for encryption, of user1's and user2's, should be different algorithm's programs. Other rows of user1 and user2 are independent.
It is clear, that the deciphering program in the row of user1, in the table of user2, must be able to decipher the enciphered data from user1.
the deciphering program in the row of user2, in the table of user1, must be able to decipher the enciphered data from user2.
For each mail address, which is appears in the table,
the enciphering program, enciphering key, deciphering program, deciphering key, stored in the reference table, being correspond to each electric mail address.
When sending e-mail, to a transmission destination electronic mail address, the e-mail address, in the table, is regarded as destination address.
the enciphering program and enciphering key in the row of this e-mail address, in the table, is used for encryption.
When receiving enciphered e-mail from a transmission source electronic mail address, the e-mail address, in the table, is regarded as source address.
the deciphering program and deciphering key, in the row of this e-mail address is used for deciphering.

the enciphering program stored in the reference table is pre-set to correspond to a transmission destination electronic mail address, and the deciphering program is pre-set to correspond to a transmission source electronic mail address.
the computer program containing instructions which upon execution carries out the following method steps:

when sending an enciphered data to an electronic mail address, this mail address determines the row of the table, from this row the enciphering program and enciphering key, are determined automatically, as this result,

automatically enciphering a file to be transmitted according to a specified a specified enciphering program upon an input of a prescribed enciphering instruction, wherein the specified enciphering program is specified by looking up the reference table by using the designated transmission destination electronic mail address as a key, and is read out from the memory of the information processing device; and

When receiving an enciphered e-mail from a transmission source electronic mail address, this source address determines the row of the table, from this row, the deciphering program and the deciphering key, are determined automatically, as this result,

automatically deciphering a received enciphered file according to a deciphering program upon an input of a prescribed deciphering instruction,
where the deciphering program is specified by looking up the reference table using the transmission source electronic mail address as a key, and is read out from the memory of the information processing device.

９．one-on-one　について、
　基本的な特徴は、
（１）暗号通信網が、各個人がそれぞれの必要に応じて構成する拡張されたアドレス帳と、そこに登録されている暗号化部分と復号化部分に分離された暗号ソフトによって構築されているところにある。
　拡張されたアドレス帳には、様々なアルゴリズムの暗号ソフトが自由に登録できます。１００人の人と暗号メールのやり取りをする人は、暗号化のアルゴリズムとして１００種類のものを使い、復号化のアルゴリズムとして更に別の１００種類のアルゴリズムを選択して、登録できます。使われる鍵も自由に登録できます。
　暗号通信網に参加する人全員が合意しなくてはならない暗号アルゴリズムなどそんざいしないのです。

（２）各個人が使用する暗号化ソフトと復号化ソフトを自分と、自分の通信相手のそれぞれの状況（機械、法律、立場など）にあわせて独立に設定できるところにある。
　それぞれの個人が自分の拡張されたアドレス帳を管理します。２つのアドレス帳の中で関連するものが存在するのは、直接に暗号化されたデータのやり取りをする２人のアドレス帳のそれぞれ１つの行の、１つの項目だけです。送信する方の暗号化の項目と受信する側の復号化の項目が関連するだけです。
　もちろん、ＡさんがＢさんに送信するときの暗号アルゴリズムと、ＡさんがＢさんから通信を受信するときの暗号アルゴリズムは異なるものが使用できるし、そうすべきです。
おなじアルゴリズムで大量のデータの送受信をすることは暗号解読の手助けとなってしまいます。
　２つのアドレス帳が関連をもつのは、直接暗号データを交換する２人の間でだけであり、さらに、暗号アルゴリズムは、二人のうちのどちらからどちらに向けて送られたかと言うことで、使用する暗号化アルゴリズムが決まる。
　つまり、
１．暗号通信をする二人が暗号通信の方式を自由に取り決めできる。(one-on-one)
２．さらに、信号の向かう方向ごとに暗号アルゴリズムが決まる。(ordered pair)
となることが私の方法の特徴です。

（３）全体としての暗号ソフトのデータベースなどは使用しない。
（共通なデータベースを使用すると全体との関係に縛られて自由に更新したり、高度化したり出来なくなる。）

（４）細かな設定が可能となるのは、暗号化と復号化を分離したからです。これによって、ＡさんからＢさんへの呼びかけを暗号化するアルゴリズムと、ＢさんのＡさんに対する応答を暗号化するアルゴリズムを異なったものにすることができる。これを、電子メールの暗号化にも、電話による暗号通話にも使うことができると考えています。

（５）これによって、経済的に余裕のある人は常に暗号通信の安全性を強化できる。古い機械を使い続けることも可能である。無駄なデータを蓄積した暗号ソフトのデータベースは必要ない。暗号製作者の著作権を守り経済的基盤を提供できる。国ごとに異なる法律の壁を越えた暗号通信が違法とされること無く実現できる。

これらが、one-on-one と ordered pair で表現しようとしたところの本来の意味です。

　アドバイスに従って、提出した文書の　ｐ.8　を修正いたします。
p.8 lines 19-26
The claimed matter is such as to enable automatic determination of an enciphering program for each transmission destination, while allowing use of different enciphering program for each destination.
The foundation of this system is, a one-on-one relationship between the sender and receiver, decide the use of enciphered communication.
Further more there is tow direction of data stream, from A to B, from B to A. According to the direction, we can use enciphering programs of different algorithms, such as DES, and RSA. The algorithm is decided according to the direction of data stream, According to the order of (A to B), (B to A), decide the algorithm independently each other. This is the key to in the claimed arrangement.
Such cannot be achieved by the disclosure of Ichikawa since the enciphered packet is designated to be decoded by more than one terminal. Thus, the claimed subject matter is distinctively different from that of the reference Ichikawa, and therefore the present invention cannot be considered obvious from Ichikaw even if combined with Ueno.

１０．２－６についても意味が明確になるように１２項で修正いたします。
１１．特になし

１２．ここでは、ネットワームを流れるデータの構造について、述べています。
電子メールに関しては初期のインターネットでは、単純なテキストしか送信や受信が出来なかった。バイナリーデータを送信できるようにするために、ＲＦＣ822 によって、MIME 形式が認められた。
　バイナリーデータを base64 エンコードによって変換することで、インターネットを通じてどんなデータでも送信や受信ができるようになった。
　電子メールでは、添付ファイルとして指定されたものは、どのようは形式のデータであってもバイナリーデータとしての扱いを受けて、ネットワーク内を流れてゆく。

暗号化されたデータは、バイナリーデータと見て扱うのが適当であり、電子メールにおいては添付ファイルとして扱えばよい。
音声データを暗号化した場合も、既存の通信網の通信規約を侵害しないように注意して扱わなくてはなりません。

これを述べているのが　claim2　です。
クレーム２の書き換えは、
2. A computer program according to Claim 1, further comprising electronic mail transmitting means for transmitting the enciphered file, this is must be treated as binary data file, to be transmitted as attached file to an electronic mail, to the designated transmission destination.
Binary file is translated by base64 encode, this translation make it possible to sending the binary file to the Internet.
In the case of phone, the enciphered data must be translated to a suitable form, for the protocol of transmitting data in the network.

これは、通信規約内で送信することを述べているので当たりまえのことかもしれません。

１３．クレーム３、５について
クレーム２には、application order data の記述はありません。この記述はクレーム３とクレーム５に存在します。
クレーム３では、テーブルの行が、アドレスによって決定されたとき、その行の左から右に向かって登録されている複数の暗号化ソフト、複数の暗号化鍵、複数の復号化ソフト、複数の復号化鍵に適用するときの番号を付けることができることを示しています。
とくに、多重暗号化の場合は、この番号に従って暗号化され、逆の順番で復号化されることになります。
異なる点は、Ａさんのアドレス帳での、Ｂさんに暗号データを送るときの暗号化のソフトの摘要順と、Ｂさんから暗号化されたデータを受け取るときの復号化ソフトの摘要順は独立である。送信時は左から右、受信時は右から左のように。

このこと注意して、クレーム３を書き換えれば
3. A computer program according to Claim 1, wherein;
said the row of reference table, is configured to have the plurality of enciphering programs and plurality of enciphering keys, corresponding to a single transmission destination electronic mail address.
and the application order data of these encryption programs to the data to be transmitted, is pre-set.
wherein the file to be transmitted is enciphered in multiple stage using a number of the plurality of enciphering programs in a sequence based on the pre-set application order data; and

said, the row of reference table is configured to have the plurality of deciphering programs and plurality of deciphering keys, corresponding to a single transmission source electronic mail address,
and the application order data of these deciphering programs to the enciphered data to be decrypted, is pre-set.
wherein the enciphered file, specified by the transmission source electronic mail address, is deciphered in multiple stage using a number of the plurality of deciphering programs, in a sequence based on the pre-set application order data.

For an electronic mail address in the reference table, the application order data for enciphering, and the application order data for deciphering, are independently pre-set.

In the case A send a data to B, the application order data for enciphering programs in A's table, and the application order data for deciphering programs in B's table, are reverse order.
In the case A receive a data from B, the application order data for deciphering programs in A's table, and the application order data for enciphering programs in B's table, are reverse order.
But these two application order data for enciphering programs of A and B, are independent.

クレーム５の書き換えは

5. A computer program according to Claim 3,
In a reference table, a row of an electronic mail address has plurality of enciphering programs,
wherein the application order data of enciphering programs is automatically changed suitably according to a pre-set rule.
In the same reference table, a row of an electronic mail address has plurality of deciphering programs,
wherein the application order data of deciphering programs is automatically changed suitably according to another pre-set rule.
These two pre-set rules, for encryption and decryption, in the same row, of the table, are independent.
These two application order data, for encryption and decryption, in the same row, of the table, are independent too.

The pre-set rules are independent for each electronic mail address.

１４．１５．

クレーム４修正について

4. A computer program according to Claim 3, wherein said reference table is configured to have the plurality of enciphering programs for each transmission destination electronic mail address, and the plurality of deciphering programs for each transmission source electronic mail address.

We pre-set subsets of enciphering programs, and the application order of enciphering programs in the subset, corresponding to the transmission destination electronic mail address,
one subset of enciphering programs are applied to the main text of the electronic mail, and another subset of enciphering programs are applied to an attached file of the electronic mail.
This is same to decide for each enciphering programs, is it applied to main text or not, and, is it applied to an attached file or not.

We pre-set subsets of deciphering programs, and the application order of deciphering programs in the subset, corresponding to the transmission source electronic mail address,
one subset of deciphering programs are applied to the enciphered main text of the electronic mail, and another subset of deciphering programs are applied to an enciphered attached file of the electronic mail.
This is same to decide for each deciphering programs, is it applied to enciphered main text or not, and is it applied to enciphered attached file or not.

For each transmission destination electronic mail address or transmission source electronic mail address,

１６．

ここでは、テーブル構造が不明確であり、電話での呼びかけと、その呼びかけに対する応答では、異なるアルゴリズムの暗号化が使われていることが不明確になっています。
従って、ヨーロッパのものに手を加えて修正案を作ります。

クレーム６の修正は以下のようになります。

6. A telephone machine having a transmitter notification means; comprising,
means for installing an enciphering and a deciphering program;
memory means for storing an enciphering reference table in which a transmission destination telephone number is pre-set to correspond to an enciphering program;
memory means for storing a deciphering reference table in which a transmission source telephone number is pre-set to correspond to a deciphering program;

means for enciphering and transmitting a voice signal to be transmitted using the enciphering program, wherein said enciphering program to be used for the communication is identified based on the

transmission destination telephone number of the communication, and the enciphering reference table; and

means for deciphering and audibly outputting the received voice signal using the deciphering program, wherein said deciphering program to be used, for the communication, is identified based on the transmission source telephone number of the communication, is specified in the deciphering reference table.

When A call to B, A say to B "How are you?", this is enciphered by the enciphering program which is in the A's telephone machine. This enciphering program is decided using the telephone number of B, and enciphering reference table. In this case B's telephone number is regarded as transmission destination telephone number.

B answers to A "Fine thank you." this is encrypted in B's telephone machine and deciphered in the telephone machine of A. In this case the enciphering program is decided using the telephone number of A and B's enciphering reference table. The deciphering program is decided using B's telephone number and deciphering reference table of A. In this case the B's telephone number regarded as transmission source telephone number.

The enciphering algorithm of A and enciphering algorithm of B is independent.

クレーム７は、ヨーロッパ特許庁を使って

7. A telephone machine according to Claim 6, wherein;

said enciphering reference table is enabled to have a plurality of kinds of enciphering programs corresponding to a signal transmission destination telephone number, wherein the voice signal to be transmitted to the transmission destination has corresponding enciphering programs determined, said means for enciphering is enable to encipher in multiple stage, the voice signal to be transmitted using a plurality of kinds of enciphering programs; and

said deciphering reference table is enabled to have a plurality of kinds of deciphering programs corresponding to a single transmission source telephone number,

wherein the enciphered voice signal transmitted from the transmission source telephone number, has corresponding deciphering programs determined and said means for deciphering is enable to decipher in multiple stages the enciphered voice signal using a plurality of kinds of deciphering programs.

１７．最初に、Vidrascu､Leppek, Collins, Schier の論点を整理し、それらを総合したとき、どこまでが普通に考えつくことなのかを述べたいと思います。

Vidrascu の特徴は、IP アドレスと IP アドレスの間での暗号通信を提起している。安全な通信網と安全でない通信網を分けて考えている。扱っている暗号方式は１つだけであり、さらに、送信時に受信側で復号化に使用する鍵の存在を確認している。

Leppek の特徴は、多重暗号化を扱っているが、暗号ソフトのデータベースを使用した方式となっている。使用する暗号ソフトを決定するアルゴリズムが不明確であり、このとき使用する鍵が何かは明記されていない。

Collins の特徴は、SMS アドレスとインターネットメールアドレスの対応を扱っている。

Schier の特徴は、テーブル８０のコピーを共有する点にある。もちろん、鍵の番号のところを、電話番号に置き換える発想は思いつけるかも知れない。しかし、これでは、電話番号と暗号化アルゴリズムの対応が得られるだけである。さらに、コピーを共有することは、全員が同じ表を持つことになる。この表に登録されるのはすべての電話番号となってしまう。こんな表は無意味でであるし、現実的ではない。

さて、Collins の示したことから、インターネットメールアドレスと電話番号を対応させることを考えつくことは可能であろう。

さらに、Vidrascu の IP アドレスをインターネットメールアドレスに置き換えることも困難ではありません。また、暗号化鍵のこともそんなに扱いが困難だとは思いません。

Leppek の多重暗号化を使えば、インターネットメールアドレスに対して、そのアドレスに送信するデータを多重暗号化できるように、対応表を作る事はそれほど困難とは思いません。
Schier の表の鍵番号のところを、電話番号に置き換える発想は思いつけるかも知れない。しかし、これでは、電話番号と暗号化アルゴリズムの対応が得られるだけである。さらに、コピーを共有することは、全員が同じ表を持つことになる。この表に登録されるのはすべての電話番号となってしまう。こんな表は無意味でであるし、現実的ではない。
したがって、メールアドレス、暗号化ソフト、暗号化鍵の対応表までが普通に考えつく事柄であると思います。暗号データベースを共有すると言う発想も Vidrascu、Schier
では共通である。

ＡさんからＢさんへ暗号化データを送信するときの様子と、それを受け取ったＢさんが復号化するときの様子に関しては記載されているが、
欠けているのは、ＡさんがＢさんから受け取ったものを処理するときの記述である。これが欠ける理由は、暗号化データベースでは、ＢさんがＡさんから送られてきたものを扱う時と同じであると思っているからです。つまり、暗号通信を対称的なものとして考えているからである。
これは、全く別である。暗号通信を対称なものと考えるか、非対称的なものと考えるかの違いである。
非対称的な暗号通信だけが、この解明を必要とする。非対称的であるからこそ、ＡさんがＢさんから受け取るときの処理に関する解明が必要となる。
考察は４つ必要なのである。
１．ＡさんがＢさんに送信するとき。
２．ＢさんがＡさんから送られてきたデータを処理するとき。
３．ＢさんがＡさんに送信するとき。
４．ＡさんがＢさんから送られてきたデータを処理するとき。
少なくとも、この４点に関して考察しなくてはならない。
１と２のみの考察で済ませて、残りは暗黙のうちに仮定した通信の対称性を暗黙の前提とするがゆえに、不十分な考察となっているのである。
Ａさんの手元で行なわれる、１，４の対比が可能になるには、
アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵
の考察が無くてはならない。
この考察が、暗号通信の非対称性を導き、暗号データベースの破棄を実現してくれます。これが、様々な問題を解決してくれるのです。

しかし、もっとも困難なのは、
メールアドレスに、暗号化ソフト、暗号化鍵のほかに、復号化ソフト、復号化鍵を対応させること。そして、共通の暗号データベースを決して作らないことです。
この対応が様々な問題を解決してくれる様子をご覧いただければ、一見単純に見える復号化部分の追加が、なぜ重要なのかが理解できて、それを支える問題意識の違いが明確になると思います。そして、共通の暗号データベースを共有しないことが、国際的な通信にとってなぜ重要化を確認すれば、見かけ上の僅かな違いが解決する問題はとてもおおきい事がわかります。
解決したいと思うことと、技術は深い関係を持ちます。普通に考え付くか否かは解決すべき課題を普通に考えつくか否かと同じであると考えます。私の方法は沢山の問題点を解決し、強力な暗号通信網を実現できる方法であり進歩した方法であると確信しています。

詳細は順に見てゆくことになりますが、
ここで概略を簡単に見ておきます。
最初に、Vidrascu の問題点を確認します。
Vidrascu は彼自身の暗号通信の方法を提起している。しかし、Vidrascu の方法は沢山の問題点を抱えていて、とても安全な通信の方法を与えてくれるものではない。

以下問題点を列挙する。
１．暗号化されない信号が流れる経路の長さが最短ではない。
安全とされるネットワークの中では暗号化されないデータがながれ、安全ではないとされるネットワークとの境目で暗号化されるのです。安全とされるネットワークの存在を仮定しているところがもっとも大きな

問題点である。組織内部の人間から情報が漏洩する事が多いのは現代の情報犯罪の特徴である。

２．沢山のデータが同一のアルゴリズムを使って暗号化をされる。
　電子メールの仕組みにおいては、沢山のメールアドレスが、同一の IP アドレスに対応する。一つのメールサーバーを使う沢山の人たちは、同一の IP アドレスを使っている。したがって、IP アドレスと IP アドレスの対応をもとに暗号通信網を構築すると、沢山の電子メールが同一の方法で暗号化されることになる。これは、暗号解読の大きな手がかりとなる。

３．盗聴者も自分の作ったデータをその共通の方法で暗号化できる。
　盗聴者が、目的とするメールサーバーの利用者として自分を登録すれば、暗号解読に役立つデータを送信し、アルゴリズムの特徴をとらえて、解読に成功する可能性が高くなる。

４．使用する暗号技術の決定に問題がある。
　バックドアがあり簡単に解読できるような暗号ソフトが使われているか否かの判断は難しい。サーバー管理者が選んだ方法が安心できないことも十分考えられる。
　新しい暗号技術が開発されたとき、メールサーバーを管理して、そこを通過するデータを暗号化しようとする人は、開発者にソフトの代金を幾ら払うのであろうか？
　メールサーバーの数はそんなに多くはない。少数の暗号ソフトだけが採用されることになる。開発者の開発費は十分に補填されないであろう。
　更なる問題は、暗号の安全性に誰が責任を持つのだろうか？国や組織によっては、暗号化鍵、復号化鍵を提出するように要求するところもある。この情報が漏れたりしたら、暗号通信の意味が無くなる。

５．使用する暗号技術への変更に問題がある。
　新しい技術を使用するとき、その解読方法がすでに存在するならば暗号通信がむいみなものになる。
　その有効性を利用者にきちんと証明できるのでしょうか？長い間、バックドアの存在を疑われていた暗号アルゴリズムも存在した。証明付きで安全と言われていた方法が解読されてしまった例もある。
　古くなった暗号ソフトを、高い金を払って更新して、３年も経たないうちに解読だれたら誰が責任を取るのであろうか？更新するときの費用は誰が負担するのであろうか？

６．国による法律の違いに細かな対応が出来ない。
　国や組織によっては、暗号メールを受け取れるが、暗号メールを送信できないような規制があったり、登録された暗号化鍵や復号化鍵しか使えなかったり、弱い暗号化しかつかえなかったりする。
　暗号通信網に参加する人がそれぞれの通信相手ごとに、きめ細かく暗号化の設定や暗号化をしないような設定が可能でなくてはならない。IP アドレスから IP アドレスへの対応では、このことは不可能である。

以上の問題点は、私が提案する方法で解決できる。
１．パソコンから出るときに、データは暗号化されている。また、暗号化されたデータはパソコンの中で復号化される。したがって、暗号化されないデータが流れる通信経路は存在しない。

２．沢山のデータが異なるアルゴリズムを使って暗号化される。
　私が提案する方法では、経済的、技術的な条件が許せば、それぞれの送信者のメールアドレスと受信者のメールアドレスの組に対して異なる暗号化アルゴリズムを使用することができる。
　しかも、A さんが B さんに送信するときの暗号化アルゴリズムと、A さんが B さんから受信したデータが暗号化されているアルゴリズムは異なるものであってもかまわない。
　したがって、同一の方法で暗号化されたデータの量は極めて少ない。

３．盗聴者は自分の作ったデータをその方法で暗号化できない。
　盗聴者が、通信経路に対象とする方法で暗号化された様々なデータを流してみるのは不可能である。
　暗号化の方法が不明。分かっているなら、データを流してテストする必要は無い。

４．使用する暗号技術の決定が個人の判断でできる。
　新しい暗号技術が開発されたとき、各個人がその内容を調べて信頼できる優れた技術ならすぐに採用すればよい。不安なら自作すればよい。責任は、自己責任である。
　各個人が優れた暗号ソフトを購入するなら、開発者の生活も保障され、さらに新しい暗号を開発できる。

５．使用する暗号技術への変更が自由にできる。

信頼できない暗号技術はすぐに捨てて、新しいものに自由に変更できる。

６．国による法律の違いに細かな対応が出来る。
国や組織によっては、暗号メールを受け取れるが、暗号メールを送信できないような規制があったり、登録された暗号化鍵や復号化鍵しか使えなかったり、弱い暗号化しかつかえなかったりする。
暗号通信網に参加する人がそれぞれの通信相手ごとに、きめ細かく暗号化の設定や暗号化をしないような設定が可能でなくてはならない。
メールアドレスごとに、送信の場合と、受信の場合で別々に設定できるから、通信相手の住んでいる国の法律に合った方法で暗号通信を利用できる。

これらが可能となったのは、拡張されたアドレス帳が、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の項目を持っていることと、拡張されたアドレス帳を各人が独自に管理できるからである。
これにあわせて、暗号化部分と復号化部分に分離された暗号ソフトを使うことによって強固な暗号通信網を構築できる。決して共通の暗号データベースを持たないことが重要である。

詳細は以下の項目への反論の中で明らかとなってゆきます。

17. At first order the assertions of Vidrascu、Leppek, Collins and Schier. I will show the combined result and difference of my method.

Vidrascu asserts the encrypted communication between IP address and IP address. His assumption is the existence of safety network. The encryption is done at the end point of the safety network. He is treating only one ciphering algorithm DES. He verifies the existence of two keys.

Leppek asserts multiply enciphered communication, which using database of cipher programs. The key is not clear, and the algorithm is not clear for deciding the program, which to be used.

Collins asserts the correspondence and transformation of SMS address and Internet mail address.

Schier asserts two persons should have the copy of the table.

For changing these conception, it will be easy to change IP address of Vidrascu to e-mail address, change the number of Schier to e-mail address, and adding multiply encryption of Leppek, we can get the table, which have columns of address and encryption programs. And the addition of columns of encryption keys may be easy.
But adding the columns of decryption programs and decryption keys are not easy. This is contradicting to using the database of cipher programs. Leppek and Schier using database of cipher programs. Without give up the using of the database of programs, we can not add the columns of decryption program and decryption key.
All of these assertions having the table of tow columns, number and encrypt program.
This is the result of the assumption of symmetry of the communication. They treat the description of enciphering at A and decryption at B.
We must consider the enciphering at A and decryption at B and enciphering at B and decryption at A. The decryption at B is inverse of encryption at A. The encryption at B is inverse of decryption at A.
A can treat two things, encryption at A and decryption at A. As this consequence we can get the structure of the extended address book.

No one says, the existence of database is fundamental reason of problems for the international communication.
The technique is the result of awareness of the problems. I have removed the database of programs, from the ciphered communication. As this result, the columns of decryption programs and decryption keys are needed.
It may be possible to use same algorithm for sending and receiving. In the case using the database, the assumption of using same algorithm is used.
This assumption of same algorithm must be repealed. For constructing international ciphered communication, and for strong cipher network.
My extended address book is the answer to these problems.

At here I will confirmation of each literature.

Vidrascu's problem

1. The length of the route of the flowing plane data is not minimum length.

He assuming the existence of safety network and not safety network. Encryption is done at the boundary of these networks.

Where is safety network? All most of the flow out of important information are inside leak. The assumption of safety network must be throw away.

2. Too many data are enciphered using same algorithm.

He decides the algorithm according to the pair of IP addresses. In the case of e-mail mail server has IP address. Many e-mail addresses using same IP address. As the result of this, too many data are encrypted using same algorithm.

3. Wiretapping person can flow his special data to this network.

For decoding ciphered data, wiretapper collects many data. It is clear, if wiretaper can flow his special data to this cipher network, he will get much more advantage.

The provider of Internet, can not distinguish whether his customer is wiretapper or not.

4. How decide the algorithm to be used?

It is difficult to judge the existence of the backdoor of the cipher algorithm. The algorithm, which is selected by the caretaker of the mail server, may be not safety.

Who can accept responsibility? In the case of submission of keys to the government, can we continue this cipher network.

5. How change the algorithm.

There is many cipher programs, which became to trash. If a new cipher program is adopted, and it is immediately decoded. Who should pay the cost of changing the crypt software. Who can probe the safety of new software.

6. The problem of corresponding to the law of each government.

The restrictions in each country are very different. Receiving the ciphered data is possible, but sending ciphered data is not allowed. The key must be registered. The length of the key must be short, etc.

The agreement on the software, which should be used, will not reach to a point. What algorithm's program should be used, what key should be used.

My extended address book gives us the solutions for these problems.

1. When the data is flow out from a computer, it is enciphered. After the enciphered data reached a computer, the data is decrypted. There is not exists the route of flowing plane data.

2. Many algorithms are used for enciphering data.

My extended address book gives us the possibility, different crypt algorithm can be used to each address. And the two algorithms may be different, from A to B and from B to A.

3. Wiretapper can not encrypt his special data using other person's cipher program.

Each person can use different algorithm. There is not exists common cipher program. Wiretapper can not know the algorithms of other person.

4. Each person can decide to use or not use of a new cipher program.

When new algorithm is given, each person can test it and decide the usage of it, on the responsibility of him self.

Each person pays the price of the program. The developer of cipher program can get the profit. This is the foundation of new and powerful program.

5. Changing is easy.

If the method of decoding is appears, user can change the cipher algorithm.

6. Correspondence to the laws of each country.
Each person should consider the condition of his partner, who is living another country.
Whether he can receive ciphered data or not. Whether he can send a ciphered data or not. For these conditions must be reflected on to the extended address book.

The columns of address, encrypting program, encrypting key, decrypting program and deciphering key, give us the settlement of these problems.
The extended address book is the opposition of database of cipher programs.

１８．私が提案する方法では、通信を行なう二人を考えたとき、公開鍵暗号方式で RSA と ECC が使われるとすれば、一方の人が持っているのは、RSA の暗号化ソフトと、RSA の公開鍵、ECC の復号化ソフトと、ECC の秘密鍵、相手の人が持っているのは、RSA の復号化ソフトと、RSA の秘密鍵、ECC の暗号化ソフトと、ECC の公開鍵であり、鍵は二人のものを合計すると４つになります。
もちろん、これら４つの鍵が同時に存在する必要は無いし、その一部しか存在しないこともあります。暗号化したデータを送信するときには、送信者として使用する暗号化ソフトと暗号化鍵があることを確認するだけで十分です。受信者が復号化のための鍵を持っているか否かは確認の必要がありません。これが、国境を越える暗号化通信を合法的に行なえる根拠になります。さらに、受信者が復号化の鍵を持っているかどうかを、送信するときに確認する必要はありません。受信者は、暗号化されたままのデータを受け取っておいて後で復号化すればそれで済むからです。
18. When considering the communication of two persons, A uses RSA for encrypting the data from A to B, B uses ECC for encrypting the data from B to A.
A has the key for encryption of RSA, and the key for decryption of ECC. B has the key for decryption of RSA, and the key for encryption of ECC. We must consider the existence of these four keys.
When A sending the data, the existence of enciphering program and enciphering key must exist. But the existence of deciphering program and deciphering key in B's address book, will not be confirmed.
When A receives the data, the deciphering program and deciphering key must exist. But the existence of encryption program and encryption key in B's address book will not be confirmed.
If B can not send enciphered data, then the decrypting program and decrypting key are not exist in the row of B in the A's address book.
This non-symmetrical structure of the extended address book is the foundation of the international ciphered communication.

Vidrascu　col.6 line21-31
When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.
The transmission is next sent to the interface 31;
When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is　decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30.

Vidrascu が col.6 line21-31 で述べている暗号化プログラムと復号化プログラムは、
暗号化プログラムがインターフェイス３０にあり、復号化プログラムはインターフェイス３１にあって、インターフェイス３０で暗号かされたものを、受信したインターフェイス３１で復号化する場合の話であって、存在する場所が違う。
インターフェイス３０の所に、暗号化プログラムと復号化プログラムを備えるようにして、その対応について考えているのではない。暗号化されたデータは、受信されてから復号化されるのは当然のことだから、この

考えからは、表の項目に、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵を加えなくてはならない必然性は出てこない。
しかも､暗号化方式として DES しか考えていないから､暗号化と復号化で異なるアルゴリズムを使ったり、送信のときは暗号化して、受信は暗号化しないような発想は全く含まれていない。
DES は対称鍵方式なので鍵は共通である。送信時でも２つの鍵を調べている。受信時でも２つの鍵を調べている。これは、A から B への送信が DES 方式で暗号化されるとき、B から A への返信も２つの鍵が存在するのだから同じく DES で暗号化されることを意味している。しかも同じ鍵を使って暗号化されるのだから、同じ鍵が手元に２つ存在することを調べても意味は無い。
必要なのは、
A が B に送信するときに必要となる暗号化ソフトと暗号化鍵を確認することである。存在すれば暗号化して送信し、存在しなければ平文のまま送ればよい。
B からのデータを受信するときには、B に対応する復号化ソフトと復号化鍵の存在を調べることである。存在しないときは平文のままで送られてくるとみなせばよい。
A が B に暗号文を送れても、B は A に平文しか送れないこともあるのだから。このような状況に対する配慮が無いから、インターフェイス３０に、暗号化ソフトと復号化ソフトの両方があるかどうかを調べようとしないのである。
もちろん、この二つは異なるアルゴリズムの暗号化ソフトと復号化ソフトであると考えなくてはならない。

別の問題もある。
user@domein 形式のメールが扱えるようになったのは１９８６年です。ＩＰアドレスは@の後ろの domein に対応します。したがって、
user11@domein1 と user12@domein1 は同じＩＰアドレスを持ったメールサーバーを使い、
user21@domein2 と user22@domein2 の使うメールサーバーも同じＩＰアドレスを持つことになります。
したがって、
user11@domein1 から user21@domein2 への通信と
user12@domein1 から user22@domein2 への通信では同じ暗号化が行なわれます。
同じプロバイダーを利用する人はとても多く、同じＩＰアドレスを使うユーザーが必然的に多くなります。よって、同じ方式、同じ暗号化鍵で大量のデータを暗号化して送信することになります。こらは暗号が解読されてしまう大きな原因になります。
ＩＰアドレスとＩＰアドレスの間の対応はメールサーバーとメールサーバーの間の対応を意味しています。これでは暗号ソフトを変更できるのはメールサーバーの管理者だけになってしまいます。同じプロバイダーのメールサーバーを利用している人たちはたくさんいます。そのうちの２つのメールサーバーを利用している人たちのグループは同じ暗号化ソフトを使うことになります。これでは暗号ソフトを、電子メールの各ユーザーが自由に変更することは出来ません。解読技術が見つかったらすぐに変更できなくてはなりません。
電子メールのユーザーが自分が信頼できる暗号ソフトを選んで自由に利用するためには、ＩＰアドレスとＩＰアドレスの対応ではなく、メールアドレスとメールアドレスの間の対応に従って暗号化ソフトや暗号化鍵を設定できなくてはなりません。自分が新しい機械を購入したらより強固な暗号に変更出来るようにしておくべきです。これが出来なければ法律的な制限が異なる外国に住んでいる人との暗号通信がトラブルの元になってしまいます。自分のアドレス帳に登録されている様々な国の人々と円滑な暗号通信を実現するには電子メールの利用者が自分のアドレス帳の中で自由に、きめ細かく設定できなくてはなりません。
さらに、暗号の鍵をＩＰアドレスに合わせて選ぶことは書かれているが、暗号ソフトをＩＰアドレスにあわせて選択するとは書かれていない。この中で現れる暗号化ソフトはＤＥＳだけであり、同じ相手似たいする送信時と、同じ相手からの受信時で異なるアルゴリズムの暗号を使うような記述は存在しない。
さらに、復号化ソフト、復号化鍵を相手の IP アドレスにあわせて選択するとは書かれていない。

この方式は更なる問題点を含んでいます。安全なネットワークと安全でないネットワークを分けて考えて、その境界で暗号化と復号化を行なおうとしている点です。問題は、安全とされるローカルなネットワークが存在するか否かです。会社の顧客情報は高い値段で売り買いされます。社員が安全とされるローカルなネットワーク内で暗号化されないで流れているパケットを盗聴し外部に販売する可能性も十分に考えられます。情報漏洩のほとんどは社内の人間が機密情報を社外に流出させるような形です。したがって、安全なネットワークと安全でないネットワークの境界など存在しません。自分のパソコンから出る時には暗号化されている必要があるのです。従って、暗号ソフトの管理は各パソコンで直接しなくてはならないし、各人が自分のパソコンの中に拡張されたアドレス帳を持っている必要があるのです。

Vidrascu explains, at col. 6 lines 21-31, encrypting process at interface 30, and decrypting process at interface 31. But he is not explains the decryption at interface 30.

In the case only one algorithm is used in the ciphered network, it may be obvious the same algorithm is used.

But many algorithms are used in the ciphered network, it is obvious to use same algorithms for sending and receiving. If we consider the restriction of the law, we must consider the two cases of enciphering and deciphering at one side.

This is the reason, extended address book having the columns of enciphering programs and deciphering programs.

If Vidrascu is considering encryption at interface 30 and decryption at interface 30. He will be able to get same result. But he is not explains these things.

His method has more weakness. He is treating DES. DES's encryption key and decryption key are same.

I can not understand, why he wants to verify the two keys for encryption and decryption, they are same. And if two keys are existing at interface 30, it is not mean the existence of the key at interface 31.

More over, in the case of open key system as RSA, the key for decryption is never given to the sender of ciphered data. We can not see these two keys at interface 30.

We don't need verify the existence of decryption key, when sending data. We should verify the existence of encrypting program and encrypting key, when sending data.

If it is exist, send an enciphered data. If it is not exist, send a plane data.

There is another problem.

From the year 1986, we are using the e-mail address of the type user@domain.

IP address corresponding to the "domain". As this result user11@domain1 and user12@domain1 are use same mail server of an IP address. user21@domain2 and user22@domain2 use same mail server of another IP address.

If the encryption key is decided from the pair of IP address and IP address, same encryption key is used with the communication from user11 to user21, the communication from user11 to user22, the communication from user12 to user21 and the communication from user12 to user22.

Many persons use same mail server. As this result huge data is encrypted, using same program and same key. This is the leap of decoding.

The relation of IP address and IP address means the relation of mail server and mail server. This fact means only the manager of the server can change the crypt program.

The user of the mail server can not change the cipher program. Every user should be able to change cipher program. Only the reliable cipher programs should be used. For realize this idea, we should use the correspondence of e-mail address and e-mail address instead of correspondence of IP address and IP address.

For the international communication, the structure of extended address book is needed, the columns of deciphering program and deciphering key must be exist.

We must throw away the database of cipher programs. If each member living different countries. It is impossible to distribute the common database to the member of the encrypted communication network.

There is another problem.

He believes the existence of safety network. No one can give the assurance of the local network. When the data flow out from the computer, it must be enciphered.

col.12　line 1-11 では

if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.

verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and

if the first and second keys exist, deciphering a received message with the second key and otherwise reject

a received message.
　ここで確認しているのは、鍵の存在だけで暗号ソフトの存在を確認してはいない。もし暗号ソフトを何種類も使ったり、暗号化する場合と暗号化しない場合があるのなら、必ず確認しなくてはならない項目である。その記述が無いのは、暗号ソフトはＤＥＳのみを考えていて、暗号ソフトの種類を、送信のとき（たとえばＤＥＳ）と受信のとき（たとえばＡＥＳ）で変更しようと言うことは、まったく考えていないからです。
　暗号ソフトの変更を考えるなら、誰がどのような立場で決定し、どのようにして暗号ソフトの変更や登録をするかについての記述が必要です。
　keys　と複数形になっている事が問題です。送信と受信を分離するなら、送信の暗号化に必要な鍵は１つす。送信のときは暗号化鍵と暗号化ソフトの存在だけを考えればよい。
　暗号化されたデータを受け取れても、暗号化されたデーターを送信できない立場の人もいるのですから。

　私の方法では、送信の時に確認するのは、暗号化鍵と暗号化ソフトです。暗号データを受け取っても、暗号化されないデータのみを返してくる人もいるのだから。
　受信時には、復号化鍵と復号化ソフトの存在だけを考えるのが当然である。その人から、暗号化されたデータを受け取れても、その人に対して暗号化されたデータを送信できない場合もあるからです。暗号化されたデータを受け取っただけで犯罪者扱いされてしまう国の存在も考えられます。
　暗号メールは受け取れないが、暗号メールを送信できる場合も考えられます。したがって、送信のときは暗号化鍵と暗号化ソフトだけを、受信のときは復号化鍵と復号化ソフトだけを調べるようにしないと、国境を越えた暗号通信が出来なくなります。
　この文献では、暗号プログラムをユーザーが自由に登録、変更が可能となるよう薬方法はまったく示されていません。メールサーバーで一括管理では、プライバシーを守れません。

At here, he verifies only the existence of key, he does not verify the existence of program. If he is thinking to use many kinds of crypt programs, he should talk about the existence of the program.
He is verifying the existence of two keys. This is most different part to my idea, and the part, which is never understandable for me. If the crypt program is DES only, two keys are same. Why he wants to verify the existence of two keys?
The second key is used at the destination. For verify the existence of the second key at the destination, we must call to the destination. This is contradict for the nature of e-mail. The existence of the second key at the source, is not mean the existence at destination.
When using open key system, the second key must not to be exhibited. In this case the source side can not verify the existence of the second key.

We must consider the four keys, it is not necessary all of these key's existence. The situation of the people are different each other.

1. A person can send encrypted data, and he can receive encrypted data.
2. A person can send encrypted data, but he can not receive encrypted data.
3. A person can not send encrypted data, but he can receive encrypted data.
4. A person can not send encrypted data, and he can not receive encrypted data.

We must consider and correspond to these cases. For the international network cipher communication, the address book must be extended, and the database of cipher programs must be discarded.

　Vidrascu の方法にはさらに大きな問題点があります。
col. 6　lines 23-27 では
if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.

Fig. 13 95, 102
1st　ans 2nd keys exist?

Fig. 13 97
Encipherment of message to be sent by 1st key

Fig. 13 104
Decipherment of received message with 2nd key

この鍵２がどこに存在することを、どのようにして確認するのだろうか？
（１）送信者の手元に２つともあることを確認するのであろうか？
（２）送信者の手元に鍵１があり、受信者の手元に鍵２があることを別の通信で確認するのであろうか？
（２）の方法では、沢山の IP アドレスに向かってデータを送信する場合に現実的な意味を持たなくなってしまいます。
（１）の場合は、もっと問題が大きくなる。共通鍵方式では、暗号化鍵も復号化鍵も同じである。同じものが２つあったことが、受信者がその鍵を今持っていることの保証にはならない。公開鍵方式では、受信者は彼の持つ秘密鍵を他人に渡してはならない。送信者が持っているのは受信者が作成した公開鍵だけでなくてはならない。送信者が復号化で使用する秘密鍵を持っているような方法は公開鍵方式での暗号通信を台無しにしてしまう。これは、ばかげた方法である。

このことからも、私が提案する方法は Vidrascu の提案する方法とはことなり、現代の暗号化方式の中で重要となっている公開鍵方式に対応できる優れた方法である。
Vidrascu の欠点は、暗号ソフトや鍵をどのような形式で登録すべきかを研究していないところにある。

暗号化された音声信号は、復号化された後で再度音声に変換されなくてはならないことは誰が考えても明らかです。
しかし、呼びかけのときの暗号方式と、それに答えるときの暗号方式を別のものにしようと考えたのは大きな進歩だと思います。
これが、暗号化された音声は送信できないが、暗号化された音声の受信が可能となっている国に住む人と、自由に暗号化した電話を利用できる国に住む人との合法的な暗号通信の基礎を与えてくれる。
送信時に使用できる暗号化アルゴリズムや、鍵の長さに制限がある国に住む人と、制限の無い国に住む人の通信では、暗号アルゴリズムと鍵を、受信時と送信時で全く独立に設定できなくてはならない。

Vidrascu's method has more problems.
Where exist the two keys? How can we verify the existence of these keys.
(1) Does the sender verify the existence of two keys in his hand?
(2) Does the sender verify the first key in his hand, and verify the second key is in the recipient's hand.

Case (2) has problem. Is it possible to verify the existence of second key?
Should sender phone to recipient? The answer of recipient may be a lie. The e-mail is sent to many places of the world. The verification is impossible.

Case (1), has two problems.
First problem is that he is not verifying the existence of the key at recipient. The key is used at recipient. If sender having the second key, the recipient may not having the second key.
Second problem is that, when open key system is used, the second key is never exhibited. Sender can not verify the second key.

As the result, Vidrascu's method is not applicable to the open key system. My extended address book is applicable to the open key system.

The defect of Vidrascu's system is that, he did not analyze the way of registration of enciphering program, enciphering key, deciphering program, and deciphering key.

It is ordinary that enciphered voice signal must be decrypted and changed to voice signal.
At first I thought that the enciphering algorithm of call and deciphering algorithm of answer should be different.
This method gives me a suitable system for international cipher network.

Scheir , Fig.3 and 6 について
この図の中には、（電話番号として解釈できるところの）鍵と、暗号化ソフトの対応しか書かれていない。暗

号化鍵の部分が必要であり、さらに、復号化ソフト、復号化鍵の項目が無くてはならない。これらの項目が無ければ、電子メールの利用者一人一人が自由に暗号ソフトを利用することは出来ないし、国ごとの法律の違いや、組織ごとの規則の違いにきめ細かく対応できない。

Scheir , Fig.3 and 6
In this figure, first column is key, second column is program.
There is not the column of enciphering key, deciphering program and deciphering key.
These columns are the foundation of the freedom for each person, deciding the algorithms to be used. According to the difference of law of each country, we must set the programs and keys. This is possible on the extended address book. It is not possible, when using the database of cipher programs.

Scheir　col. 10 line 1 で
receiving device 78 includes a copy of table 80
と言っている。
　これは、全体が共通の暗号ソフトデータベースを持つことを意味していて、著作権問題、法律問題、各人の経済的問題などに対応できない方法であることを示している。
　さらに、通信相手がコピーを持つということは、この暗号通信網に盗聴者が参加できるなら盗聴者もコピーをもつことになる。これは暗号通信網の有効性を自ら否定することになる。
　拡張されたアドレス帳は、決してコピーされてはならない。暗号通信網全体としての暗号アルゴリズムの多様性と、各個人に集積される暗号アルゴリズムを必要最小限にとどめることが、通信網全体の安全性にとって重要である。Scheir の方法では不可能だが、私の方法ならば、これが可能となる。
Scheir col. 10 line 1
He says
receiving device 78 includes a copy of table 80
This means all the member of this cipher network, having same copy. This is the cause of many trouble copyright, patent right, economy (rich and poor) and law.
In the case the wiretapper entry to this cipher network he will have same database and he will be able to reverse assemble.
The extended address book should not be copied. The diversity of algorithms is the foundation of the safety. Non accumulation of cipher programs is a foundation of safety too. My system allows using the cipher program, which is made by user him self.

Schier , col.9 line67　col.10 line1-7 では
The receiving device 78 includes a copy of table 80. The copy of table 80 is then used at step 132 to retrieve the first indicated encryption algorithm from the encryption algorithm column 84. This encryption algorithm is then loaded into the encryption decryption engine 30 and is executed by central processing unit 20 to encrypt outgoing communication and decrypt incoming communication at step 134.
　この問題点は、テーブル 80 のコピーを持っていることである。テーブルに記載された暗号化アルゴリズムの番号によって指定される暗号化ソフトと同じ番号で指定される復号化ソフトが 78 と 72 の両方に存在することである。
　このことは、ＡさんとＢさんが同じテーブルを持つなら、ＢさんとＣさんも同じテーブルをもち、。。。となって、全員が同じテーブルを持つことになります。
　暗号通信網に参加する人全員が、共通のテーブルを持つのは幾つかの点で問題を含みます。

１．このテーブルは追加、削除、更新ができるのか？
２．追加、削除、更新の決定は誰がするのか？
３．追加したソフトはどのようにして配布するのか？
４．著作権料はだれが、どのように払うのか？
５．国を超えた配布は犯罪行為にならないか？

６．テーブルが新しい強力な暗号に変わったとき、古い機械の持ち主は利用できるか？
７．新しい機械を持っている人は、使えない古い暗号を保存する必要がなぜあるのか？

テーブルを共有するとこのような問題を突きつけられてしまいます。テーブルの共有をあきらめなくてはなりません。逆に、決して共有しないことが、強固な暗号通信網を構築する基本原理なのです。
各人が拡張されたアドレス帳を自分の分だけ管理することが、上記の矛盾を解決してくれます。

コピーを使うということは、この暗号通信網に参加するすべての機械が同じテーブルと同じ暗号化ソフト、同じ復号化ソフトを共有することになる。
この暗号化通信網が、旧式の機械の参加を認めるからには低い計算能力でも実用となる弱い暗号化を使うことになってしまう。
すべての機械がコピーを持つからには、その暗号化通信網に参加する機械の台数が多くなると、新しい暗号アルゴリズムに変更することが極めて困難になる。なぜなら、暗号化ソフトや復号化ソフトの著作権料を払える人と払えない人がいるからであり、また異なる国の法律はそのアルゴリズムの使用を許す場合と、許さない場合がある。強力な暗号ソフトを国外に持ち出すことを禁止している国もあるし、自由に持ち出すことを許している国もある。コピーをインターネットで配布すると犯罪行為になってしまう事も多い。
ある国では、受信するデータの暗号化については問題にしないが、送信するデータの暗号化には制限をかける場合も考えられます。
そうすると、Ａさんが住んでいる国では自由に暗号が使え、Ｂさんが住んでいる国では受信は自由だが送信するときは簡単な暗号化しか許されない場合もあります。このとき
Ａさんのアドレス帳の暗号化ソフトは最新の強力なものを登録し、暗号化鍵も長いものを
登録できます。しかし、Ｂさんのところの復号化ソフトは簡単なものでなくてはならないし、また復号化鍵の長さは、短いものでなくてはなりません。
Ａさんの暗号化のテーブルと暗号ソフトをＢさんに送ってしまったら、Ｂさんがそれを使っただけで犯罪者になってしまいます。暗号に関して、法律的な制限が異なるのが現実です。この中で少しでも有効な暗号通信網をつくるには、各人が独自に管理する拡張されたアドレス帳（電話番号帳）が必要なのです。
さらに、復号化鍵を国の公共機関へ提出することを求めている国もある。コピーを使うならば、鍵を提出させた国はすべての通信を解読することができる。このような集積された復号化鍵、暗号化鍵の情報は高い値段で販売できるので、公共機関内部からの情報漏洩の心配も起こる。また、ネットワークに参加する人全員が暗号化と復号化を共有するなら全員が解読する能力をもつことになってしまう。
以上の理由で、この方法は国際化された通信には向かない。コピーをあきらめなくては
ならない。しかし、私が拡張されたアドレス帳の考えを使えばこの困難は解決できる。
単に電話番号から暗号化ソフトを呼び出すだけなら簡単な話であるが、暗号ソフトを暗号化部分と復号化部分に分離し、それに合う形でアドレス帳を拡張し暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵を登録できるようにしたことがもっとも重要な点であると確信しています。この点についてはまったく記述されていない。コピーを配るような記述は自ら暗号通信網を破壊するようなものであり、決して行なってはならない。

私の方法では、
ＡさんとＢさんが新しい電話機を買って、Ｃさんが古い電話機を使いつづけているとします。このとき、私の提案する方式では、Ａさんの持っているアドレス帳のＢさんの行に登録される暗号ソフトと暗号化鍵は最新のものを登録してそれを使うことが出来ます。
Ｃさんは古い電話機しか持っていないので、Ａさんと暗号通信を行なうときもＢさんと暗号通信を行なうときも簡単な暗号化方式を使わないます。古い電話機では復号化や暗号化に時間がかかりすぎてしまうので、新しい強固な暗号を使うことはできません。したがって、Ｃさんの拡張されたアドレス帳のＢさんの行には古い簡単な暗号ソフトが登録されていなくてはなりません。
したがって、Ａさんの拡張されたアドレス帳とＣさんの拡張されたアドレス帳は待ったくべつのものでありコピーではありません。
各個人が自分の持っている電話機の性能と自分の通信相手の持っている電話機の性能に合わせてそれぞれ暗号化、復号化の設定をできるようにしておくほうが、暗号通信網全体としての強度は向上します。
この点から見ても、アドレス帳のあるいは電話番号帳の項目に、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の５つの項目が必要となるのです。

各人のアドレス帳（暗号表）は各人で管理する。共通化はしない。これが大切である。こうすれば新しい機械がすぐに力を発揮できる環境を作れます。古い機械が暗号通信網の安全性を脅かしてはいけない。各人が自分と自分の通信相手の状況にあった形で拡張されたアドレス帳の内容を調整できることが大切です。

この点から見ても、拡張されたアドレス帳は暗号通信の安全性を飛躍的に改善し、しかも国際暗号通信での法律的な問題点も解決できる方法であり、従来の技術よりも優れたものであると確信しています。

これが可能となったのは、ＡさんとＢさんの会話を、ＡさんからＢさんへの方向性を持った情報伝達と、ＢさんからＡさんへの方向性を持った情報伝達の２つに分けてそれぞれ別々の暗号化アルゴリズムで暗号化しようと考えたからです。

記号化すれば、ＡとＢの会話｛Ａ、Ｂ｝を（Ａ→Ｂ）と（Ｂ→Ａ）の２つに分解して、それぞれに対して別々の暗号化を導入したということです。

Schier, col.9 line67 col.10 line1-7

Device 78 having the copy of table 80, this is important problem. If A and B having copy of the table, B and C must having the copy of the same table, and so on. As this result all person have common table.

Many troubles appear from this sharing.

1. Addition, deletion, renewal are difficult
2. Who decide Addition, deletion and renewal.
3. Who can distribute the database?
4. How pay the cost.
5. The distribution of this table and the programs to be used is difficult.
6. The old machine obstacle the efficiency.
7. The new machine should saving the old software and terms in the table.

The database of crypt programs has more important problems.

The first problem is

1. Addition, deletion, renewal are difficult.

The standard of crypt algorithm is changing fast. The new algorithm needs powerful computer. But there is the user of old computer. When adding new one, the user of old computer must keep a program, which is never used. When deleting old program, the user of old computer can not participate to this network. The renewal bring the destruction of the network.

The second problem is

2. Who decide Addition, deletion and renewal.

It is difficult that all the people in network agree to use a new algorithm. It is difficult to prove there is no back door.

The third problem is

3. Who can distribute the new programs to be added to the database?

There is strong restriction on the export of crypt programs. In the case, the network uses the common database of cipher programs. Who can distribute the new program without guilty decision of judge?

The upload of cipher programs to the Internet, become to crime in some country.

If it is possible in a country, he must pay the cost of program, it must be distributed to all the user of this system. It is too high for a person.

The fourth problem is

4. How pay the cost.

The problem is copyright and patent right.

In the case, every one has the common database, how much the price of database?

The database will include many programs, which will not be used. The person must pay the price of the software, which is never used. The new participant will not use the old software, because he will have new machine.

In the case of adding new crypt program to this database, dose every one agree to this?

If this program is new and need powerful CPU, the person who has old computer, will not agree to add this program. He must pay the cost of software, which he never uses.

In this cases, the person who has old computer, should be excluded or not. This is the reason of breakdown of cipher network.

The database must be unique and all members must have it. When adding new program, the distribution of the copy is difficult. If it is powerful, we can not distribute it using Internet. In the case, using post for distribution of program, who should pay the cost of the post.

The programs are progressing from DES, Triple-DES to AES and RSA to ECC. We should not use the database of crypt programs.

For the development of new crypt program, much more time and much more cost. The user must pay the cost. This is the foundation of new and powerful crypt program.

The fifth problem is
5. The distribution of this table and the programs to be used is difficult.
There is strong restriction on the export of crypt programs. In the case, the network uses the common database of cipher programs. Who can distribute the database of cipher programs, without guilty decision of judge?
The upload of cipher programs to the Internet, become to crime. No one can construct crypt network, without guilty decision of judge.
Very weak crypt programs may be able to export. But the total database of cipher programs must exist in each person's computer.
For this reason, the database of cipher program can not construct cipher network on the Internet.

The sixth problem is
6. The old machine obstacle the efficiency.
If using database, every one must keep old programs for the user of old machine.

The seventh problem is
7. The new machine should saving the old software and terms in the table.
If using database, every one must keep old programs for the user of old machine.

The restriction of law.
The legislation in each country is different very much.
In a country, it is possible to send ciphered data without any restriction. Another country allows accept encrypted data, but forbid send an encrypted data.
There may be the country whish has contrary system.
A country forbids sending and receiving ciphered data.
A country allows ciphered data accepting and sending, with the condition of submission of encryption key and decryption key.
The internationalization of communication is normal communication. The shearing of common database of crypt programs may make a criminal.
In the case of a country, there is restriction of the length of key, appropriate length key must be used in the communication.
In the case of a country, the exhibition of key is duty, who is living in the country should exhibit the keys, which he has.
If the database is used, the outflow all of the keys may occur. This is a weak point of database of crypt programs.
In this point of view, the unification of database of crypt programs must be discarded.
Each person must use crypt programs within the law. The total safety of the crypt communication network must be protected.
Each person should keep least programs and keys, which is needed, in his computer. It must be possible to manage his extended address book. This condition gives us the safety of total cipher network.
In this case, if a person's computer cracked, the repair is easy. Only the person and his friends rewrite their address book, and change the crypt programs.
By my method the influence of submission of keys become minimum. If A is living a country where he can use cipher free. And B is living a country, where he must submit the key. A should use simple cipher program for the communication with B. A should use powerful cipher program for the communication with other person.
It is the reason of the safety of the cipher network, don't accumulate cipher programs. If a person get a new machine, who can change his cipher program.

The second problem is export.
There is more strong restriction on the export of crypt programs. In the case, the network uses the common database of cipher programs. Who can distribute the database of cipher programs, without guilty decision of judge?
The upload of cipher programs to the Internet, become to crime. No one can construct crypt network,

without guilty decision of judge.

Very weak crypt programs may be able to export. But the total database of cipher programs must exist in each person's computer.

For this reason, the database of cipher program can not construct cipher network on the Internet.

Extended address book can solve this problem. Each person can select the cipher programs for registration into his address book. Considering the conditions of his friends.

The third problem is copyright and patent right.

In the case, every one has the common database, how much the price of database?

The database will include many programs, which will not be used. The person must pay the price of the software, which is never used. The new participant will not use the old software, because he will have new machine.

In the case of adding new crypt program to this database, dose every one agree to this?

If this program is new and need powerful CPU, the person who has old computer, will not agree to add this program. He must pay the cost of software, which he never use.

In this cases, the person who has old computer, should be excluded or not. This is the reason of

In this cases, the person who has old computer, should be excluded or not. This is the reason of breakdown of cipher network.

The database must be unique and all members must have it. When adding new program, the distribution of the copy is difficult. If it is powerful, we can not distribute it using Internet. In the case, using post for distribution of program, who should pay the cost of the post.

The programs are progressing from DES, Triple-DES to AES and RSA to ECC. We should not use the database of crypt programs.

For the development of new crypt program, much more time and much more cost. The user must pay the cost. This is the foundation of new and powerful crypt program.

My extended address book and separated crypt program make it possible. Every user can buy new program and distributes encryption part and key to his friend, and register the decryption part and key.

It is clear safety RSA need long key too old computer can not correspond to this long key.

An old computer in the net work, using database, is the cause of weakness of this net work.

In the case of telephone machine, sending enciphered data. The improvement of the machine is progressing day by day.

New machine is powerful, but only few persons can purchase it, as soon as it appears.

Poor person uses old machine for a long time.

Old machine can treat simple cipher program. It is not appropriate to use powerful encryption algorithm.

If new powerful program is used on the old machine, too long time is needed for encryption and decryption.

We consider the case, three persons A, B and C are the members of cipher communication network. The person A and B have new machine and C has old machine.

Suppose that, A send encrypted data to B and C. All of B's friends having new machine.

B does not send data to C.

For the communication of A and C, C can not use powerful cipher programs. If A want to use new cipher program to the communication with C, it is impossible for C's machine can not decrypt it in an appropriate time.

In the case, using database of crypt program, the participation of C is cause of futile. All most of the member of this crypt network having new machines, the preservation of old program is waste of memory.

The exclusion of C gives efficient network. This is destruction of network. The database of crypt programs is the reason of this destruction.

We should repeal the database of crypt program. My extended address book will be able to give the system, which does not use the database of crypt programs.

Each person should control his own address book. This gives us most efficient cipher network. The member possesses least and powerful crypt programs. Who has powerful machine, and all his friends has powerful machine, he can use powerful programs only.

In A's address book, the row of C, contains simple enciphering program, short enciphering key, simple deciphering program and short deciphering key. In A's address book, the row of B, contains powerful enciphering program, long enciphering key, powerful deciphering program and long deciphering key.

In C's address book, the row of A, have simple enciphering program, and simple deciphering program.

The row of B does not exist in C's address book

In B's address book, all of the programs are powerful and new.
As the result the table of C is not copy of the table of A. The address book A, B and C are different.
The idea of database of crypt program is the reason of many troubles. Only the extended address book gives us the real construction of crypt network. And give us the harmony of old machine and new machine.

From these points of view, my extended address book gives us powerful and safety cipher network, according to the law.
This is originate from the fact, the communication of A and B, must be considered the communication from A to B, and the communication from B to A. I have introduced different encryption algorithms to these two communication.

１９． Schier , Fig.3 and col. 4 lines 30-56 では
FIG. 3 illustrates a selected portion of one embodiment of encryption table 28 which was disclosed with reference to FIG. 2 previously. Encryption selection table 28 comprises a key column 34, a first algorithm column 36, a second algorithm column 38 and a third algorithm column 40. In operation, a particular device as device 14 would have the capability of performing a number of distinct encryption processes. For example, device 14 may be able to perform five different encryption techniques. An encryption key in key column 34 is then used to access a particular row which specifies a particular encryption technique in each of columns 36, 38 and 40. A message to be sent by device 14 or received by device 14 can then be encrypted or decrypted using the technique specified in the row in the order specified in the row. For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4. In this manner, a number of relatively simple encryption steps can be sequentially applied to a message to greatly enhance the security of the message. A person attempting to intercept and wrongfully decrypt the message would have to discern not only the various kinds encryption used, but also the order in which the technique were used.

問題点は
For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1.
Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.
の記述にある。
device 14 における暗号化アルゴリズムと device 14 における復号化アルゴリズムを分けないのである。これは暗号通信の対称性を無条件に認めていることを示している。
暗号データを受け取れても、平文しか返せない人や、強力な暗号を施したデータを受け取れても弱い暗号化をしたデータしか返せない人の存在を無視している。
誰もが同じ立場ではない。暗号化と復号化では異なるアルゴリズムを採用できる仕組みを用意しなくてはならない。
法律の問題は、共通の暗号ソフトを持つ事を許さない場合が多い。暗号通信が犯罪者を作り出してはいけない。
対応テーブルに、復号化部分を独立した形で設けることが必要なのです。 Schier の方法はこの問題を解決できない。

Schier says

For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1.
Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14

would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.

He says at the device 14, the algorithms are same of encryption and decryption. He is assuming the symmetry of cipher network. But we must consider that there is a person who can receive enciphered data but he can not send enciphered data.

As this result at device 14, we should consider different algorithms are used, for sending and receiving.

Law may forbid to sending encrypted data and allow to receiving encrypted data.

For this reason we need extended address book, which having the columns of address, encrypting program, encrypting key, decrypting program and decrypting key.

２０．Vidrascu の方法は、たとえ各個人が IP アドレスを持っていると解釈しても、Fig. 13 での、２つの鍵の確認をすることや、鍵や暗号化ソフトを登録するテーブル構造が明記されていないなど、

現代の公開鍵暗号方式には全く対応できないものであり、時代遅れの方式であると言わなくてはならない。

このようなものが、安全で合法的な暗号通信網を与えてくれないことは明らかであり、確実に強固で合法的な暗号通信網を構築できる私の方法とは全く異質のものである。したがって、Vidrascu の示した内容が、私の発明の拒絶理由になるとは思えない。

このような違いを生み出す根本は、拡張されたアドレス帳を各個人が持っているか否か、さらに、その中の項目として、

名前、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵

の項目が存在するか否かと言うところにある。

この点が私の提案する方式のもっとも核心的な部分である。Vidrascu はこの点については何ら解明していない。

The method of Vidrascu is out of date. He is verifying the two keys at Fig. 13. This method is mean less and opposite to open key system as RSA. He does not explain the structure of the table.

Even if we interpret the IP address as e-mail address, he does not explain the necessity of the columns of address, enciphering program, enciphering key, deciphering program and deciphering key.

This structure gives us legality cipher network. Each person can management his address book, without the database of crypt programs.

２１．Vidrascu、Leppek, Collins, Schier の方法を統合すれば、メールアドレスから暗号化の鍵を決定したり、電話番号から多重暗号化で使う一連の暗号ソフトを決定したりするようにすることを思いつくのはそんなに困難ではない。これは審査官の言われる通りです。

彼らは、A とＢの通信を考えるときに、A がＢに送るものを暗号化する過程の説明をします。そして、受け取ったＢが復号化するときの様子を解説します。

彼らが使う対応表は、A が暗号化で利用する表と、Ｂが復号化で利用する表の２つです。

さらに、考察しなくては成らないことが２つあります。それは、Ｂが A に送るものを暗号化するために使う表と、A がＢから受け取ったものを復号化するために使う表です。

これを考えなければ、対応表の中に、復号化ソフト、復号化鍵の部分をいれて、A がＢに送るものを暗号化する時に使用するものと、A がＢから送られてきたものを復号化するときに使用するものを意識的に分けて入れると言う発想は生まれません。

これを、別々に登録するからには、異なったアルゴリズムのプログラムを使用できなければ意味がありません。従ってこの場合の暗号通信は非対称となります。さらに、暗号通信網で使われる暗号化プログラムがすべて異なるアルゴリズムでもかまわないと言うことが理論的な可能性として出てきます。これは、暗号ソフトのデータベースを使う意味が無いということを教えてくれます。

これは、決して普通に思いつく事柄ではない。対称性や暗号データベースを捨てないと思いつかない事柄で

す。したがって、Vidrascu、Leppek, Collins, Schier の方法を寄せ集めても、拡張されたアドレス帳の構造は出て来ません。
　現存する暗号通信で暗号データベースを使ったり、対称的な暗号通信の性質を前提としたりすることが、沢山の問題を含んでいることに気づき、これらを解決するという強い意志が無ければ、拡張されたアドレス帳の発想は生まれてこない。
　単に、項目を増やしただけではあるが、この構造こそが暗号通信網に内在する問題点を一挙に解決してくれるのです。したがって、発明としての意義は十分あると確信しています。
　以下詳しく見てゆきます。

Integrating the methods of Vidrascu, Leppek, Collins and Schier, it is not so difficult to get the table, which have the columns of address (phone number) and encryption programs.
When considering cipher communication of A and B. They explain the process of enciphering at A. And they explain the process of deciphering at B.
We must consider more two cases, we must explain the process of enciphering at B. And we must explain the process of deciphering at A.
From this consideration, we can get the structure of the table which having the columns of address, enciphering program, enciphering key, deciphering program and deciphering key.
This means the usage of different algorithms in the communication from A to B and in the communication from B to A.
As the result of this, the non-symmetric cipher communication appears. It is obvious, in logically, all the enciphering programs used in cipher network, have different algorithms. This means the database of programs is not needed.
I think it is difficult to discard the database. As the result, it is difficult to think of the extended address book.
Integrating the methods of Vidrascu, Leppek, Collins and Schier, we can not get the structure of extended address book.
When using database, we can not think of the table which having the columns of address, enciphering program, enciphering key, deciphering program and deciphering key.
I know the database and symmetric structure of cipher communication, is the cause of many troubles. I know that for solving these problems, we must discard database and symmetry in cipher communication.
My extended address book is my answer to these problems. I believe that the will to the settlement of these problems, appeared as extended address book. My system can solve many problems of cipher network.
Details are as follows.

Vidrascu, col.6 lines 21-27
When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.
The transmission is next sent to the interface 31;
　最初に問題となるのは IP アドレスの意味である。これが暗号通信網に参加する個人を特定できるか否かである。
　Vidrascu の文献が提出されたのは 1993 年であり、電子メールに関しては、
　user@domein 形式のメールが扱えるようになったのが１９８６年です。ＩＰアドレスは@の後ろの domein に対応します。したがって、
user11@domein1 と user12@domein1 は同じＩＰアドレスを持ったメールサーバーを使い、
user21@domein2 と user22@domein2 の使うメールサーバーも同じＩＰアドレスを持つことになります。
　したがって、
user11@domein1 から user21@domein2 への通信と
user12@domein1 から user22@domein2 への通信では同じ暗号化が行なわれます。
　同じプロバイダーのメールサーバーを利用する人はとても多く、同じＩＰアドレスを使うユーザーが必然的に多くなります。よって、同じ方式、同じ暗号化鍵で大量のデータを暗号化して送信することになります。

こらは暗号が解読されてしまう大きな原因になります。
　ＩＰアドレスとＩＰアドレスの間の対応はメールサーバーとメールサーバーの間の対応を意味しています。これでは暗号ソフトを変更できるのはメールサーバーの管理者だけになってしまいます。同じプロバイダーのメールサーバーを利用している人たちはたくさんいます。そのうちの２つのメールサーバーを利用している人たちのグループは同じ暗号化ソフトを使うことになります。これでは暗号ソフトを、電子メールの各ユーザーが自由に変更することは出来ません。解読技術が見つかったらすぐに変更できなくてはなりません。
　電子メールのユーザーが自分にあった暗号ソフトを選んで自由に利用するためには、ＩＰアドレスとＩＰアドレスの対応ではなく、メールアドレスとメールアドレスの間の対応に従って暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵を自由に設定できなくてはなりません。
　これが出来なければ法律的な制限が異なる外国に住んでいる人との暗号通信がトラブルの元になってしまいます。自分のアドレス帳に登録されている様々な国の人々と円滑な暗号通信を実現するには電子メールの利用者が自分のアドレス帳の中で自由に、きめ細かく設定できなくてはなりません。
　さらに、問題となるのは、
if the keys related to the IP addresses of the sender and of the receiver are found,
の部分です。公開鍵方式を考えると、暗号化の為の公開鍵と復号化の為の秘密鍵を両方とも送信者側のメールサーバー管理者の手にゆだねることになります。もちろん、秘密鍵は、暗号化データを受け取りそれを復号化する大切なものですから、各個人の手元に置いておかなくてはなりません。もちろん、受信側となるメールサーバーに置くことも出来ません。暗号化された情報の最終的な受信者以外の手元に置いてはいけません。公開鍵暗号のいみが無くなります。
　IP アドレスが各個人を表すと考えても、同様です。公開鍵暗号を考えると、暗号化で使う公開鍵だけを確認すればよいのであって、秘密鍵まで確認するようなシステムは完全に時代遅れです。
　これを確実に実現できるようにするためには、拡張されたアドレス帳に、名前、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の項目を作らなくてはなりません。
　しかも、暗号化ソフトと暗号化鍵は登録されているが、復号化ソフトと復号化鍵は登録されていないような状況でもきちんと動くシステムを作らなくてはなりません。
　もちろん、逆の状況すなわち、暗号化ソフトと暗号化鍵は登録されていないが、復号化ソフトと復号化鍵は登録されているような状況でもきちんと動くシステムを作らなくてはなりません。
　さらに、暗号化ソフトのアルゴリズムと、復号化ソフトのアルゴリズムが全く異なるような状況でもきちんと動くシステムを作らなくてはなりません。
　これを実現するには、上記の拡張されたアドレス帳の項目が必要になるのです。もちろん、名前の項目は論理的には必要ありませんが、実用的なシステムにはなりません。

Vidrascu wrote his papers 1993.
From the year 1986, we are using the e-mail address of the type user@domain.
IP address corresponding to the "domain". As this result user11@domain1 and user12@domain1 are use same mail server of an IP address. user21@domain2 and user22@domain2 use same mail server of another IP address.
If the encryption key is decided from the pair of IP address and IP address, same encryption key is used with the communication from user11 to user21, the communication from user11 to user22, the communication from user12 to user21 and the communication from user12 to user22.
Many persons use same mail server. As this result huge data is encrypted, using same program and same key. This is the leap of decoding.

The relation of IP address and IP address means the relation of mail server and mail server. This fact means only the manager of the server can change the crypt program.
The user of the mail server can not change the cipher program. Every user should be able to change cipher program. Only the reliable cipher programs should be used. For realize this idea, we should use the correspondence of e-mail address and e-mail address instead of correspondence of IP address and IP address.
For the international communication, the structure of extended address book is needed, the columns of deciphering program and deciphering key must be exist.
We must throw away the database of cipher programs. If each member living different countries. It is impossible to distribute the common database to the member of the encrypted communication network.

There is another problem. He says,
if the keys related to the IP address of the sender and of the receiver are found.
This is the disturbance of using open key system as RSA. The secret key must be kept at the side of

receiver. It is clear the receiver can not put the secret key at the side of mail server.
Further more, if regarding the IP address as e-mail address, we should verify the existence of enciphering key and enciphering program.
The using of open key system is standard method. If the system can not use it, the system is out of date.
The system must move in the case of only the fellow of encryption and decryption. The system must move without verifying the existence of the second key.
The system must move in the case of using different algorithms for encryption and decryption.
Only my extended address book can solve these problems.

More problems are appears.
He believes the existence of safety network. I believe that the safety network is not exists. The one, who is inside of the organization, is the cause of the flow out of the data, in most case.
No one can give the assurance of the local network. When the data flow out from the computer, it must be enciphered.

Vidrascu, col.6 lines 27-36
When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30;
ここでも、IP アドレスの意味が、普通のメールサーバーの IP アドレスならば、サーバー管理者の手元に復号化鍵があり、サーバー管理者が個人の秘密をすべて知ってしまう事になります。
IP アドレスを、すべての個人が持っていると仮定しても、暗号化されたデータを復号化するには、復号化の鍵があれば十分です。
さらに、暗号の鍵をＩＰアドレスに合わせて選ぶことは書かれているが、暗号ソフトをＩＰアドレスにあわせて選択するとは書かれていない。この中で現れる暗号化ソフトはＤＥＳだけであり、他の暗号化アルゴリズムを選ぶような記述は存在しない。
この方式は更なる問題点を含んでいます。安全なネットワークと安全でないネットワークを分けて考えて、その境界で暗号化と復号化を行なおうとしている点です。問題は、安全とされるローカルなネットワークが存在するか否かです。会社の顧客情報は高い値段で売り買いされます。社員が安全とされるローカルなネットワーク内で暗号化されないで流れているパケットを盗聴し外部に販売する可能性も十分に考えられます。情報漏洩のほとんどは社内の人間が機密情報を社外に流出させるような形です。したがって、安全なネットワークと安全でないネットワークの境界など存在しません。自分のパソコンから出る時には暗号化されている必要があるのです。従って、暗号ソフトの管理は各パソコンで直接しなくてはならないし、各人が自分のパソコンの中に拡張されたアドレス帳を持っている必要があるのです。

If the manager of the server has the key of the decryption and program for decryption, he knows all secrets of communications.
At interface 31, when deciphering the received encrypted data, only use the key for decryption. The encryption key need not exist at interface 31 If we assume that each person has the IP address of him self.
He explains that selection of key according to the IP address. He does not explain the selection of encrypting program. He is treating DES only.
This system includes further problem.
He is believing the existence of the safety network. He want to do the encryption and decryption at the boundary of safety network and non-safety network.
All most of the outflow of information is inside job. The information of customer may be sold high price. The insider may catch the packet flowing safety network and sell it.
I believe that safety network is not exist. All the data flowing network must be encrypted.
Each person must have his extended address book. Each person must be able to manage his own cipher programs.

Vidrascu, col.6 lines 21-36
審査官は、ここで DES しか現れないことを確認している。さらに、沢山の暗号化ソフト、復号化ソフトが現れる Leppek　の文献を引用している。
多重暗号化で、沢山の暗号化ソフトが現れたり、それに対応する沢山の復号化ソフトが扱われるのは当然のことであるが、
大切なのは、A さんが B さんに送るときに DES を使い、A さんが B さんから受け取ったものを復号化するには AES を使うというように、
暗号化項目と、復号化項目で異なるアルゴリズムの暗号化ソフトが登録できるような、拡張されたアドレス帳を考えなくてはならないのです。
これが、暗号通信網全体に共通な暗号ソフトのデータベースを不要としてくれます。
これによって、暗号ソフトを違法に輸出してしまうようなことを避けられるようになります。自分が新しい機械を購入したらより強固な暗号に、すぐに変更出来るようにしてくれます。

The examiner is confirming that Vidrascu treating DES only.
Leppek explains multiple encryption, and received data must be decrypted reverse order.
The difficult point is to use different algorithms, when A sending ciphered data to B, and A receiving data from B and decrypting it.
For example in the data flow from A to B using DES, in the data flow from B to A usig AES.
This method need the columns of address, enciphering program, enciphering key, deciphering program and deciphering key.
As the result of this the database can be throw away. Each person should treat his own table. There is no necessity to register an e-mail address, which is not used. There is no necessity to register a crypt program, which is not used. We must register cipher programs according to the laws of countries.
As the result of this, when he bought new machine, he can change programs to new one, in the limitation of law.

さらに、
Vidrascu, col.12　line 1-11 では
if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.
verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and
if the first and second keys exist, deciphering a received message with the second key and otherwise reject a received message.
となっています。ここで確認しているのは、鍵の存在だけで暗号ソフトの存在を確認してはいない。暗号ソフトを何種類も使ったり、暗号化する場合と暗号化しない場合があるのなら、必ず確認しなくてはならない項目である。その記述が無いのは、暗号ソフトはＤＥＳのみを考えていて、暗号ソフトの変更はまったく考えていないからです。
暗号ソフトの変更を考えるなら、誰がどのような立場で決定し、どのようにして暗号ソフトの変更や登録をするかについての記述が必要です。
第１の鍵は暗号化に使われて、第２の鍵は復号化に使われます。送信者の手元に第２の鍵があることは、受信者の手元に第２の鍵があることの保証にはなりません。
さらに、公開鍵暗号の場合は、送信者が第２の鍵を持つことはありえません。従ってこの方法では公開鍵暗号に対応できません。
送信のときは暗号化鍵と暗号化ソフトの存在だけを考えればよい。受信時には、復号化鍵と復号化ソフトの存在だけを考えれば十分である。

At here, Vidrascu verifying the existence of key, he is not verifying the existence of programs.
When using plurality kinds of programs, the existence of program must be verified. He is thinking DES only, as this reason, he does not verify the existence of programs.

When thinking the change of programs, he must consider the structure of table. He must consider who has the authority of changing. The table is for the database, or not for the database. This is most difficult point.

He says "keys". The verification of second key has no mean. If sender having the second key, it dose not mean the existence of the second key at the receiver.

When using open key system, the second key is not exists in hand of the sender. According to this fact, this system can not to use open key system.

I think when sending data, we must verify the existence of encrypting program and encrypting key. When receiving data, we must verify the existence of the decrypting program and decrypting key. If they are not exist the encryption and decryption are not done.

Leppek, col.2 lines 19-23

The fundamental machine of the virtual encryption scheme of the involves the generation of a sequence of the access code, with immediately successive ones of the access code of the sequence being different from one another.

Leppek, col.2 lines 51-55,

The order of the encryptors within the sequence to which the data is applied may very as desired, and the sequence may 'toggle' or switch back and forth between the same set of encryption operators as part of its overall encryption flow.

Leppek, col.4 lines 14-17

The encryption routines 110, in and of themselves, need not be any particular type of encryption algorithm and may be conventional encryption operations, such as PGP, DES, etc. routines, as non-limiting examples.

暗号化の順番を変更する事に関して、考察します。
彼は、encr.key　が異なれば、異なる暗号化プログラムの列が指定されることを説明しています。
彼は、同じ encr.key に対して,supervisor encryption manager　の設定を変更して、異なる暗号化プログラムの列を生成するように出来ることを説明しているわけではない。

さらに、encryption operator database の更新方法などに関しては全くふれていない。
いま、３年も経てば、暗号アルゴリズムは古くなる。DES、 Triple―DES、 AES と変化してきた過程を見れば明らかである。したがって、使用するデータベースの更新について考察しなければ実用的なシステムとはならない。

私の拡張されたアドレス帳を使う方法では、異なるアドレスの行には異なるアルゴリズムの暗号ソフトを登録することが可能である。もちろん、適用する順も自由に設定できる。

新しいソフトと購入し、テーブルを直接書き換え、ソフトの暗号化部分を相手に送ることによって使用する暗号ソフトの更新が出来る。

もちろん、ある行に登録されている暗号ソフトを適用する順番も簡単に変更できる。直接テーブルを書き変えればよい。

もちろん、復号化部分に関しても変更は可能であり、しかも、暗号化部分とは独立して変更出来る。

この点から見ても、私の方法が、単純であるが優れた方法だと確信しています。データベースを使うとせっかくの順序変更もだいなしとなる。

At first consider the order of encryption.

He explains for each "encr.key" there appears different sequence of encryption programs.

But he does not explain, how to change the setting of " supervisor encryption manager　". If it is variable we can get different sequence of encryption programs for same "encr.key".

More over he does not explain the renewal of " encryption operator database ". After three years, the crypt programs become out of date. We have changed the programs from DES, Triple-DES to AES.

From this process, we must consider the renewal of database. If the renewal is impossible we can not use the database of cipher programs. I think the system using database is not practical.

My extended address book can register the contents of each row, independently. For each e-mail address,

different programs can be registered. The order of apply is changeable free for each e-mail address.
If user of my system bought new program, he can rewrite the table free. He can decide to use it, and he can send the part of encryption to his partner.
It is easy to change the order of applying. We can rewrite the table directly.
It is clear the decryption part can be changed easy. Of course this is independent to the change in encryption part.
I believe that my system is simple but powerful. This is the result of discarding of database.

Collins, Col.5 lines 43-63
The Internet e-mail gateway 240 converts the SMS message 100 to an Internet e-mail message having the format 150. To accomplish this, the Internet e-mail gateway 240 determines an Internet e-mail address for the cellular telephone 210 and for the Internet station 280. Regarding the Internet e-mail address for the cellular telephone 210, the Internet e-mail gateway 240 adds the domain name for the Internet e-mail gateway 240 to the SMS address for the cellular telephone located in the sender SMS address field 110 of the SMS message. To determine the Internet e-mail address for the Internet Station 280, the Internet e-mail gateway 240 uses a messaging service lookup table having a first column with a list of SMS addresses, where each SMS address corresponds to an Internet e-mail address in a second column of the lookup table. The Internet e-mail gateway 240 locates the SMS address of the Internet station 280 in the first column of the lookup table. The Internet e-mail address for the Internet station 280 is retrieved from the second column of the lookup table at a location corresponding to the Internet station SMS address of the first column.
ここでは、単に SMS アドレスと Internet e-mail address の変換とそれによって携帯メールをインターネットメールに変換することが記述されているだけです。これによって、狭い範囲でしか通用しなかった携帯メールが、インターネットを通して全世界に流通する仕組みを解明しています。

Collins explains the exchange of SMS address and Internet mail address.
By this transformation the mail of cellar phone, can be transported to the every where of the world.

さらに、
Schier, Fig.3
に関して、審査官はデータアクセスのスピードアップのために表を使うと言う見解を述べていますが、たとえ、この図の左の列をメールアドレスの列に変更しても拡張したアドレス帳は得られない。
これでは、拡張されたアドレス帳の構造は発見できません。なぜなら、ＡさんとＢさんの通信において、ＡさんがＢさんに呼びかけるときの信号伝達を暗号化する方法と、それに対応して、ＢさんがＡさんに答えるときの信号伝達を暗号化する方法を同じにすることも可能であり、そのほうが簡単なのになぜ、別々にしなくてはならないかが解明されていないからです。
これが原因で、先行技術は様々な問題点を含んでいて暗号通信の発展を困難なものにしてしまっているのです。暗号通信の発展を望みながらも、それを妨害してしまうような構造のものしか提供されていません。暗号通信の対称性を仮定する事と、暗号データベースを使うという発想が問題の根源です。
その対称性とデータベースを捨てることの必要性に気づくのは困難であり。これこそが私の発明の核心的な部分である。
著作権の問題、国ごとの法律の違い、各個人が所有する機械の計算能力の違い、新しく開発された暗号解読技術、暗号解読に使うコンピュータの計算能力の向上、などに対応しながら全体として強固な暗号通信網を構築するには、
暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の項目を備えた拡張されたアドレス帳を各個人が自分の持っている機械の性能と、各個人それぞれの通信相手のことだけを考えて自由に高度化できる方式こそが必要であり、これによって様々な問題を解決できます。
ＡさんとＢさんの通信に２つの異なる暗号化方式を用いると考えたからこそ、アドレス帳アドレス帳の４

項目の必要性となり、それによってソフトウエアの暗号化部分と復号化部分の分離による著作権問題や不正コピー問題の解決手段を提供できたのです。この点は大きな進歩であると確信します。
　このような論点は、審査官が示された文献のなかには、まったく解明されていません。
したがって、私の方法は特許されるのにふさわしい進歩性を持っていると思っています。

Schier, Fig.3
Schier is treating symmetrical communication, and database of programs.
If we changed the first column of key to the column of address, we can not get the extended address book.
For get the idea of extended address book, we must throw away the database and symmetric communication.
It may be more easy and natural way to use symmetrical cipher communication. But this system does not fit to the international network communication.
It may be natural to use the database of programs. But the database can not be distributed without unlawful action.
Preceding system can not propose effective system. The reason is database and symmetry.
We can use e-mail without saving the database of e-mail address in the own computer.
It is same, we can use ciphered e-mail without the database of cipher programs.
Only my system can answer to the problems of copyright, patent right, law, renewal of program and renewal of machine.
On my system, each person can increase his circumstance, as the result of this, the total cipher network are intensified.
My system is different from old system, and having the enough advance.

２２．これについては、審査官の言われるように、バイナリーデータを電子メールで送るには、添付ファイルとして扱い、base64 などの適当な変換をして送信するのは昔からの技術です。
　これを記述したのは、暗号化したデータはもとのデータと性質が異なるので通信網の既約に合致するようにしなくてはならない事を示したかったからです。電話などでは、あまり明確になっていないので、その必要性を強調したかったのです。
　データは通信網のプロトコルに適合しなければ送信不可能なのは自明のことなので、削除すべき項目かもしれません。

As examiner says, it is clear that the treatment of the data must fit to the protocol of the network.
I think so too.

２３．Leppek　の文献では、Fig. 2 における object 160 を、encr.key からどのようにして生成するかを説明していない。
　supervisory encryption manager の構造が不明確であり、鍵（encr.key）からオブジェクト 160 を生成する方法は全く記述されていない。encr.key が何なのかも説明していない。
　supervisory decryption manager の構造が不明確であり、鍵（decrypt.key）からオブジェクト 260 を生成する方法は全く記述されていない。decrypt.key が何なのかも説明していない。
　新しい参加者が、この暗号通信網に参加したとき、supervisory encryption manager　が対応できるのだろうか？
　もし対応できるなら、新しく追加した参加者に対して、どのような列 160 が生成されるのだろうか？
　新しい参加者はどのようにして、encryption operation data base に登録されている暗号ソフトを入手するのであろうか？
新しい参加者はどのようにして、decryption operation data base に登録されている暗号ソフトを入手するのであろうか？
　この方法では、著作権に関する問題や暗号ソフトの輸入規制などの問題は克服できない課題として残ってしまいます。
　共通のデータベースでは、暗号に関する国ごとの法律の違いを克服できません。したがって、共通の

encryption operation data base　を使うと言う発想が国際的な通信には適合しない方法である。
　さらに、テーブル 100 は、それ自体で大きな問題点を含んでいる。このデータベースは、暗号化ソフトと暗号化鍵の分離がなされていない。従って、暗号化鍵を含んだものとして考えたとき、データベースに登録しなくてはならない暗号ソフトの数は極端に多くなる。
　たとえば、RSA では、安全とされる鍵の長さは 2048 ビットで程度ですが、すぐに４０００ビット程度の長さが必要になります。単純に考えると 4000 ビットの鍵の個数は、２の 4000 乗、すなわち約 10 の 1200 乗個です。現在のハードディスクは１００G バイト程度です。これは、10 の 11 乗バイトです。いくら頑張っても、安全なデータベースは構築できません。
　全体としてのデータベースはあきらめて、各個人が自分が利用する部分だけを自由に管理できるものを中心に考えなくてはなりません。私の提案する拡張されたアドレス帳は、このような問題を解決してくれます。

Leppek does not explain what is the "encr.key". Leppek does not explain the structure of " supervisory encryption manager ". Leppek does not explain the mechanism of generation the sequence 160. Leppek does not explain what is the "decrypt.key". Leppek does not explain the structure of " supervisory decryption manager ". Leppek does not explain the mechanism of generation the sequence 260.
My system is simple and clear. Write to the table, by him self. The key is e-mail address or phone number in my system.
The next problem is on new participant.
Is it possible to distribute the software registered on supervisory encryption manager and supervisory encryption manager to the new participant.
If the database consist of old algorithm's program, the new participant should use these programs?
The new participant must pay the price of old programs. This means three years after, this network should be breakdown, because the speed of changing of crypt technology is very fast. The renewal of database is very difficult.
Next problem is the size of database.
This table does not has the column of encryption key, this means that program contains key. This means the size of database is too big.
We should throw away the database and we should use the extended address book.

Leppek, col. 4 lines 24-52
As described briefly above, the fundamental control factor used by the virtual encryption scheme of the invention is the fact it produces a sequence 160 of access address codes 120-i, such that immediately successive code 120-i and 120-j in the assembled code sequence are different from one another. Thus, for an arbitrary plurality N of respectively different data enciphering routine or operator entries 110-1, 100-2, 100-3,…,100-N, there will be N associated access address codes 120-1, 120-2, 120-3,…,120-N.
In accordance with the invention the supervisory encryption assembly manager 130 is supplied with an encryption driver or key 170 comprised of a sequence of M access code entries made up of K (at least two and up to all N) address code entries 120 for the encryption operators 110 stored in the database 100. M may be any number equal to or greater than two. Thus, at a minimum, address code sequence 140 would be defined by only two respectively different ones 120-i and 120-j of the N available codes, so that M would be equal to two, regardless of N. Even if N is only two, M is still unbounded, since it may comprise an alternating sequence of arbitrary length. Namely, where N=2, the database 100 would have only two entities 120-1 and 120-2. In this case, an encryption control access code sequence of length M could be generated as the alternating sequence …, 120-1, 120-2, 120-1, 120-2, 120-1, 120-2,…, up to M entries, where M>2. What is important is that the respective codes of any successive pair of codes differ from one another.
　ここでは、多重暗号化での通信の様子が述べられているに過ぎない。暗号化データベースの問題点には無頓着である。
　暗号データベースの構成方法、追加変更の方法に関して全く述べられていない。３年も経てば古くなってしまう暗号技術です。このデータベースが古くなって、使い物にならなくなったとき、この暗号通信網に新しく参加する人は、暗号データベースの更新を要求するだろう。これに答えることが困難なことはすでに述べました。
　しかも、新しい参加者は決して使うことの無い、古いソフトを自分のパソコンに置くことを拒否するだろう。

これでは、せっかく 160 の列を作っても、データベースの中身が違うので、受け取ったものを復号化することさえ出来ない。
　更新したデータベースを暗号通信網全体に配布することは、法律的な問題からとてつもなく困難な作業となる。インターネットで配布したとたんに、武器を輸出したとみなされて犯罪者となる。複製を大量に配布すると著作権、特許権を侵すことになる。
　もちろん、弱すぎて使う意味の無いようなおもちゃの暗号ソフトなら、配布できる可能性はあるが、これを配っても誰も使わない。
　この方法は、強力な暗号からなる暗号データベースを通信網全体に配布できると言う誤った前提に立ったものである。暗号データベースを作り、その中から選んで使うここことが大きな問題なのである。
　各人が、それぞれが必要な分だけを持てばよい。法律的に可能なものだけを持てばよい。暗号データベースを作ってはいけないのである。暗号技術の進歩は速い。自分の判断で、どんどん更新できる仕組みを作らなくてはならない。私が提案する拡張されたアドレス帳ならば、これらの問題を解決できて、全体として強力な暗号通信網を法律に触れないで構築できる。

At here Leppek explains the multiple encryption and the sequence of programs to be used. The examiner says the sequence of encryption and decryption must be specified as well. But he does not aware the problem of the cipher database.
He does not explain the way of construction of database. He does not explain the way of adding and renewal of database. After three years the database become antique. New member should demand the renewal of the database. It is difficult to distribute the database to the member in the world.
If new member communicate only few members, he can use new database. This means the destruction of cipher network. In this case sequence 160 has no mean.
This system is standing on the wrong hypothesis, that we can distribute the database to the member of this cipher network. We must consider the international network.
We must construct the cipher network without the database of cipher programs. The cipher network which does not having database of programs, must have extended address book.
Each person must decide the usage of cipher programs, according to his conditions.
This is possible. My extended address book, which permit to rewrite the terms of address, enciphering program, enciphering key, deciphering program and deciphering key. We can rewrite old program to new program. Of course he must bought new program, or make a new program by him self.

２４．たしかに、審査官の言われるように、暗号化の順序と、復号化の順序は逆でなくてはならないことは、自明です。
Schier は、
For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.
と書いています。
1. A encrypting the data to be sent to B.
2. B decrypting the data from A.
3. B encrypting the data to be sent to A.
4. A decrypting the data from B.
Schier は 1 と 3 を同じように考えています。これは暗号通信の対称性を仮定していることになります。
　しかし、A さんが B さんに多重暗号化されたデータを送信するときに、使用するいくつかの暗号アルゴリズムと、A さんが B さんから送られてきたデータを復号化するときに使う暗号アルゴリズムは全く別のものを選べることが必要であり、それを実現するための拡張されたアドレス帳の構造は決して自明ではありません。
　このような、拡張されたアドレス帳（多重暗号化の場合）のなかで、暗号化の部分と、復号化の部分に関してそれぞれ独立に適用する順序を変更できるところが新しい部分となります。

It is clear that the order of encryption and decryption must be inverse order.

We must consider four situations. With communication of A and B.
1. A encrypting the data to be sent to B.
2. B decrypting the data from A.
3. B encrypting the data to be sent to A.
4. A decrypting the data from B.
The preceding literature explaining case 1 and 2. But we must consider the relation of 1 and 3.

Schier says,
For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.

This means 1 and 3 are same, and he is thinking symmetrical cipher communication.

This is very difficult, if A can send ciphered data, but B can not send ciphered data.
Symmetrical situation is not natural condition.
In my method 1 and 3 are independent. In my extended address book, for same e-mail address, the program for encryption and decryption are independent. It is possible to use DES for encryption and to use AES for decryption at device 14.
The case 1 and 4 must be considered. In case 1 use DES algorithm and case 4 use AES algorithm. This is most different point of my method to the method of Schier.

As the result of this, we can change the orders of encryption programs, and change the order of decryption programs independently, in the case of multiple encryption.

２５．再度、主張します。暗号通信網の抱える様々な問題点を解決し、強固で簡潔な暗号通信網を構築できるのは、Vidrascu の方法ではなく、私が提案する方法だけです。
拡張されたアドレス帳が、絶対に必要なのです。

I say again that for solving many problems in the international cipher network, we must throw away the database of cipher programs and symmetrical cipher system.
Only my system of extended address book can make non symmetrical cipher system, without database of programs.
The possibility of renewal of each address book makes the total cipher network's renewal.

２６．ここでの指摘は２１での指摘と重複しますが、再度述べることにします。

Vidrascu, col.6 lines 21-27
When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.
The transmission is next sent to the interface 31;
最初に問題となるのは IP アドレスの意味である。これが暗号通信網に参加する個人を特定できるか否かである。
Vidrascu の文献が提出されたのは 1993 年であり、電子メールに関しては、
user@domein 形式のメールが扱えるようになったのが１９８６年です。ＩＰアドレスは＠の後ろの domein に対応します。したがって、

user11@domein1 と user12@domein1 は同じＩＰアドレスを持ったメールサーバーを使い、
user21@domein2 と user22@domein2 の使うメールサーバーも同じＩＰアドレスを持つことになります。
　したがって、
user11@domein1 から user21@domein2 への通信と
user12@domein1 から user22@domein2 への通信では同じ暗号化が行なわれます。
　同じプロバイダーのメールサーバーを利用する人はとても多く、同じＩＰアドレスを使うユーザーが必然的に多くなります。よって、同じ方式、同じ暗号化鍵で大量のデータを暗号化して送信することになります。こらは暗号が解読されてしまう大きな原因になります。
　ＩＰアドレスとＩＰアドレスの間の対応はメールサーバーとメールサーバーの間の対応を意味しています。これでは暗号ソフトを変更できるのはメールサーバーの管理者だけになってしまいます。同じプロバイダーのメールサーバーを利用している人たちはたくさんいます。そのうちの２つのメールサーバーを利用している人たちのグループは同じ暗号化ソフトを使うことになります。これでは暗号ソフトを、電子メールの各ユーザーが自由に変更することは出来ません。解読技術が見つかったらすぐに変更できなくてはなりません。
　電子メールのユーザーが自分にあった暗号ソフトを選んで自由に利用するためには、ＩＰアドレスとＩＰアドレスの対応ではなく、メールアドレスとメールアドレスの間の対応に従って暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵を自由に設定できなくてはなりません。
　これが出来なければ法律的な制限が異なる外国に住んでいる人との暗号通信がトラブルの元になってしまいます。自分のアドレス帳に登録されている様々な国の人々と円滑な暗号通信を実現するには電子メールの利用者が自分のアドレス帳の中で自由に、きめ細かく設定できなくてはなりません。
　さらに、問題となるのは、
if the keys related to the IP addresses of the sender and of the receiver are found,
の部分です。公開鍵方式を考えると、暗号化の為の公開鍵と復号化の為の秘密鍵を両方とも送信者側のメールサーバー管理者の手にゆだねることになります。もちろん、秘密鍵は、暗号化データを受け取りそれを復号化する大切なものですから、各個人の手元に置いておかなくてはなりません。もちろん、受信側となるメールサーバーに置くことも出来ません。暗号化された情報の最終的な受信者以外の手元に置いてはいけません。公開鍵暗号のいみが無くなります。
　IP アドレスが各個人を表すと考えても、同様です。公開鍵暗号を考えると、暗号化で使う公開鍵だけを確認すればよいのであって、秘密鍵まで確認するようなシステムは完全に時代遅れです。
　これを確実に実現できるようにするためには、拡張されたアドレス帳に、名前、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の項目を作らなくてはなりません。
　しかも、暗号化ソフトと暗号化鍵は登録されているが、復号化ソフトと復号化鍵は登録されていないような状況でもきちんと動くシステムを作らなくてはなりません。
　もちろん、逆の状況すなわち、暗号化ソフトと暗号化鍵は登録されていないが、復号化ソフトと復号化鍵は登録されているような状況でもきちんと動くシステムを作らなくてはなりません。
　さらに、暗号化ソフトのアルゴリズムと、復号化ソフトのアルゴリズムが全く異なるような状況でもきちんと動くシステムを作らなくてはなりません。
　これを実現するには、上記の拡張されたアドレス帳の項目が必要になるのです。もちろん、名前の項目は論理的には必要ありませんが、実用的なシステムにはなりません。

Vidrascu wrote his papers 1993.
From the year 1986, we are using the e-mail address of the type user@domain.
IP address corresponding to the "domain". As this result user11@domain1 and user12@domain1 are use same mail server of an IP address. user21@domain2 and user22@domain2 use same mail server of another IP address.
If the encryption key is decided from the pair of IP address and IP address, same encryption key is used with the communication from user11 to user21, the communication from user11 to user22, the communication from user12 to user21 and the communication from user12 to user22.
Many persons use same mail server. As this result huge data is encrypted, using same program and same key. This is the leap of decoding.

The relation of IP address and IP address means the relation of mail server and mail server. This fact means only the manager of the server can change the crypt program.
The user of the mail server can not change the cipher program. Every user should be able to change cipher program. Only the reliable cipher programs should be used. For realize this idea, we should use the correspondence of e-mail address and e-mail address instead of correspondence of IP address and IP address.
For the international communication, the structure of extended address book is needed, the columns of

deciphering program and deciphering key must be exist.
We must throw away the database of cipher programs. If each member living different countries. It is impossible to distribute the common database to the member of the encrypted communication network.

There is another problem. He says,
if the keys related to the IP address of the sender and of the receiver are found.
This is the disturbance of using open key system as RSA. The secret key must be kept at the side of receiver. It is clear the receiver can not put the secret key at the side of mail server.
Further more, if regarding the IP address as e-mail address, we should verify the existence of enciphering key and enciphering program.
The using of open key system is standard method. If the system can not use it, the system is out of date.
The system must move in the case of only the fellow of encryption and decryption. The system must move without verifying the existence of the second key.
The system must move in the case of using different algorithms for encryption and decryption.
Only my extended address book can solve these problems.

More problems are appears.
He believes the existence of safety network. I believe that the safety network is not exists. The one, who is inside of the organization, is the cause of the flow out of the data, in most case.
No one can give the assurance of the local network. When the data flow out from the computer, it must be enciphered.

Vidrascu, col.6 lines 27-36
When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30;
ここでも、IP アドレスの意味が、普通のメールサーバーの IP アドレスならば、サーバー管理者の手元に復号化鍵があり、サーバー管理者が個人の秘密をすべて知ってしまう事になります。
IP アドレスを、すべての個人が持っていると仮定しても、暗号化されたデータを複合化するには、復号化の鍵があれば十分です。
さらに、暗号の鍵をＩＰアドレスに合わせて選ぶことは書かれているが、暗号ソフトをＩＰアドレスにあわせて選択するとは書かれていない。この中で現れる暗号化ソフトはＤＥＳだけであり、他の暗号化アルゴリズムを選ぶような記述は存在しない。
この方式は更なる問題点を含んでいます。安全なネットワークと安全でないネットワークを分けて考えて、その境界で暗号化と復号化を行なおうとしている点です。問題は、安全とされるローカルなネットワークが存在するか否かです。会社の顧客情報は高い値段で売り買いされます。社員が安全とされるローカルなネットワーク内で暗号化されないで流れているパケットを盗聴し外部に販売する可能性も十分に考えられます。情報漏洩のほとんどは社内の人間が機密情報を社外に流出させるような形です。したがって、安全なネットワークと安全でないネットワークの境界など存在しません。自分のパソコンから出る時には暗号化されている必要があるのです。従って、暗号ソフトの管理は各パソコンで直接しなくてはならないし、各人が自分のパソコンの中に拡張されたアドレス帳を持っている必要があるのです。

If the manager of the server has the key of the decryption and program for decryption, he knows all secrets of communications.
At interface 31, when deciphering the received encrypted data, only use the key for decryption. The encryption key need not exist at interface 31 If we assume that each person has the IP address of him self.
He explains that selection of key according to the IP address. He does not explain the selection of encrypting program. He is treating DES only.
This system includes further problem.
He is believing the existence of the safety network. He wants to do the encryption and decryption at the boundary of safety network and non-safety network.

All most of the outflow of information is inside job. The information of customer may be sold high price. The insider may catch the packet flowing safety network and sell it.

I believe that safety network is not exist. All the data flowing network must be encrypted.

Each person must have his extended address book. Each person must be able to manage his own cipher programs.

Vidrascu, col.6 lines 21-36

審査官は、ここで DES しか現れないことを確認している。さらに、沢山の暗号化ソフト、復号化ソフトが現れる Leppek　の文献を引用している。

多重暗号化で、沢山の暗号化ソフトが現れたり、それに対応する沢山の復号化ソフトが扱われるのは当然のことであるが、

大切なのは、A さんが B さんに送るときに DES を使い、A さんが B さんから受け取ったものを復号化するには AES を使うというように、

暗号化項目と、復号化項目で異なるアルゴリズムの暗号化ソフトが登録できるような、拡張されたアドレス帳を考えなくてはならないのです。

これが、暗号通信網全体に共通な暗号ソフトのデータベースを不要としてくれます。

これによって、暗号ソフトを違法に輸出してしまうようなことを避けられるようになります。自分が新しい機械を購入したらより強固な暗号に、すぐに変更出来るようにしてくれます。

The examiner is confirming that Vidrascu treating DES only.

Leppek explains multiple encryption, and received data must be decrypted reverse order.

The difficult point is to use different algorithms, when A sending ciphered data to B, and A receiving data from B and decrypting it.

For example in the data flow from A to B using DES, in the data flow from B to A usig AES.

This method need the columns of address, enciphering program, enciphering key, deciphering program and deciphering key.

As the result of this the database can be throw away. Each person should treat his own table. There is no necessity to register an e-mail address, which is not used. There is no necessity to register a crypt program, which is not used. We must register cipher programs according to the laws of countries.

As the result of this, when he bought new machine, he can change programs to new one, in the limitation of law.

さらに、

Vidrascu, col.12　line 1-11 では

if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.

verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and

if the first and second keys exist, deciphering a received message with the second key and otherwise reject a received message.

となっています。ここで確認しているのは、鍵の存在だけで暗号ソフトの存在を確認してはいない。暗号ソフトを何種類も使ったり、暗号化する場合と暗号化しない場合があるのなら、必ず確認しなくてはならない項目である。その記述が無いのは、暗号ソフトはＤＥＳのみを考えていて、暗号ソフトの変更はまったく考えていないからです。

暗号ソフトの変更を考えるなら、誰がどのような立場で決定し、どのようにして暗号ソフトの変更や登録をするかについての記述が必要です。

第１の鍵は暗号化に使われて、第２の鍵は復号化に使われます。送信者の手元に第２の鍵があることは、受信者の手元に第２の鍵があることの保証にはなりません。

さらに、公開鍵暗号の場合は、送信者が第２の鍵を持つことはありえません。従ってこの方法では公開鍵暗

号に対応できません。
送信のときは暗号化鍵と暗号化ソフトの存在だけを考えればよい。受信時には、復号化鍵と復号化ソフトの存在だけを考えれば十分である。

At here, Vidrascu verifying the existence of key, he is not verifying the existence of programs.
When using plurality kinds of programs, the existence of program must be verified. He is thinking DES only, as this reason, he does not verify the existence of programs.
When thinking the change of programs, he must consider the structure of table. He must consider who has the authority of changing. The table is for the database, or not for the database. This is most difficult point.
He says "keys". The verification of second key has no mean. If sender having the second key, it dose not mean the existence of the second key at the receiver.
When using open key system, the second key is not exists in hand of the sender. According to this fact, this system can not to use open key system.
I think when sending data, we must verify the existence of encrypting program and encrypting key. When receiving data, we must verify the existence of the decrypting program and decrypting key. If they are not exist the encryption and decryption are not done.

Leppek, col.2 lines 19-23
The fundamental machine of the virtual encryption scheme of the involves the generation of a sequence of the access code, with immediately successive ones of the access code of the sequence being different from one another.
Leppek, col.2 lines 51-55,
The order of the encryptors within the sequence to which the data is applied may very as desired, and the sequence may 'toggle' or switch back and forth between the same set of encryption operators as part of its overall encryption flow.
Leppek, col.4 lines 14-17
The encryption routines 110, in and of themselves, need not be any particular type of encryption algorithm and may be conventional encryption operations, such as PGP, DES, etc. routines, as non-limiting examples.
暗号化の順番を変更する事に関して、考察します。
彼は、encr.key　が異なれば、異なる暗号化プログラムの列が指定されることを説明しています。
彼は、同じ encr.key に対して,supervisor encryption manager　の設定を変更して、異なる暗号化プログラムの列を生成するように出来ることを説明しているわけではない。
さらに、encryption operator database の更新方法などに関しては全くふれていない。
いま、3年も経てば、暗号アルゴリズムは古くなる。DES、 Triple―DES、 AES と変化してきた過程を見れば明らかである。したがって、使用するデータベースの更新について考察しなければ実用的なシステムとはならない。

私の拡張されたアドレス帳を使う方法では、異なるアドレスの行には異なるアルゴリズムの暗号ソフトを登録することが可能である。もちろん、適用する順も自由に設定できる。
新しいソフトと購入し、テーブルを直接書き換え、ソフトの暗号化部分を相手に送ることによって使用する暗号ソフトの更新が出来る。
もちろん、ある行に登録されている暗号ソフトを適用する順番も簡単に変更できる。直接テーブルを書き変えればよい。
もちろん、復号化部分に関しても変更は可能であり、しかも、暗号化部分とは独立して変更出来る。
この点から見ても、私の方法が、単純であるが優れた方法だと確信しています。データベースを使うとせっかくの順序変更もだいなしとなる。

At first consider the order of encryption.
He explains for each "encr.key" there appears different sequence of encryption programs.
But he does not explain, how to change the setting of " supervisor encryption manager　". If it is variable we can get different sequence of encryption programs for same "encr.key".

More over he does not explain the renewal of " encryption operator database ". After three years, the crypt programs become out of date. We have changed the programs from DES, Triple-DES to AES.
From this process, we must consider the renewal of database. If the renewal is impossible we can not use the database of cipher programs. I think the system using database is not practical.

My extended address book can register the contents of each row, independently. For each e-mail address, different programs can be registered. The order of apply is changeable free for each e-mail address.
If user of my system bought new program, he can rewrite the table free. He can decide to use it, and he can send the part of encryption to his partner.
It is easy to change the order of applying. We can rewrite the table directly.
It is clear the decryption part can be changed easy. Of course this is independent to the change in encryption part.
I believe that my system is simple but powerful. This is the result of discarding of database.

Collins, Col.5 lines 43-63
The Internet e-mail gateway 240 converts the SMS message 100 to an Internet e-mail message having the format 150. To accomplish this, the Internet e-mail gateway 240 determines an Internet e-mail address for the cellular telephone 210 and for the Internet station 280. Regarding the Internet e-mail address for the cellular telephone 210, the Internet e-mail gateway 240 adds the domain name for the Internet e-mail gateway 240 to the SMS address for the cellular telephone located in the sender SMS address field 110 of the SMS message. To determine the Internet e-mail address for the Internet Station 280, the Internet e-mail gateway 240 uses a messaging service lookup table having a first column with a list of SMS addresses, where each SMS address corresponds to an Internet e-mail address in a second column of the lookup table. The Internet e-mail gateway 240 locates the SMS address of the Internet station 280 in the first column of the lookup table. The Internet e-mail address for the Internet station 280 is retrieved from the second column of the lookup table at a location corresponding to the Internet station SMS address of the first column.
ここでは、単に SMS アドレスと Internet e-mail address の変換とそれによって携帯メールをインターネットメールに変換することが記述されているだけです。これによって、狭い範囲でしか通用しなかった携帯メールが、インターネットを通して全世界に流通する仕組みを解明しています。

Collins explains the exchange of SMS address and Internet mail address.
By this transformation the mail of cellar phone, can be transported to the every where of the world.

Keats, col.5 lines 6-8 (5-11)
Each network element has its own SID (maintained for example in a memory element under control of the SID manager) and also maintains a database, table or other suitable structure in another memory element, such as a cache, which maps TIDs to IP address for IP connections from the particular network element to another network element in the network, as described in detail below.
テーブル構造を利用してデータを変換するのは、昔からある。住所録、電話帳、様々な数表などなど。しかし、暗号通信における様々な問題点が何であり、それらを解決するには、どのような構造のテーブルが必要なのかについては何も述べられていない。
テーブルは、単にスピードアップの道具ではなく、問題解決の道具でなくてはならない。どの様な項目が必要となるかは、解決すべき課題によって決まってくる。国際的な暗号通信を実現するには、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の項目が必要なのである。

It is traditional technique, to use the table structure for transformation. Address book, phone book and price table etc are used a long time ago.
We must confirm the problems to be solved. We must decide the structure of the table, which can answer to these problems.

Table is not only the tool for speed up, it must be the tool for solving the problems. For the realization of international cipher communication, the table must have terms of address (phone number), enciphering program, enciphering key, deciphering program and deciphering key.

ここで、解決すべき問題点を確認し、それらが解決できる理由を説明します。

最初に問題点を列挙します。

１．暗号化されない信号が流れる経路の長さが最短ではない。

安全とされるネットワークの中では暗号化されないデータがながれ、安全ではないとされるネットワークとの境目で暗号化されるのです。安全とされるネットワークの存在を仮定しているところがもっとも大きな問題点である。組織内部の人間から情報が漏洩する事が多いのは現代の情報犯罪の特徴である。

２．沢山のデータが同一のアルゴリズムを使って暗号化をされる。

電子メールの仕組みにおいては、沢山のメールアドレスが、同一の IP アドレスに対応する。一つのメールサーバーを使う沢山の人たちは、同一の IP アドレスを使っている。したがって、IP アドレスと IP アドレスの対応をもとに暗号通信網を構築すると、沢山の電子メールが同一の方法で暗号化されることになる。これは、暗号解読の大きな手がかりとなる。

３．盗聴者も自分の作ったデータをその方法で暗号化できる。

盗聴者が、目的とするメールサーバーの利用者として自分を登録すれば、暗号解読に役立つデータを送信し、アルゴリズムの特徴をとらえて、解読に成功する可能性が高くなる。

４．使用する暗号技術の決定に問題がある。

新しい暗号技術が開発されたとき、メールサーバーを管理して、そこを通過するデータを暗号化しようとする人は、開発者にソフトの代金を幾ら払うのであろうか？

メールサーバーの数はそんなに多くはない。少数の暗号ソフトだけが採用されることになる。開発者の開発費は十分に補填されないであろう。

更なる問題は、暗号の安全性に誰が責任を持つのだろうか？国や組織によっては、暗号化鍵、復号化鍵を提出するように要求するところもある。この情報が漏れたりしたら、暗号通信の意味が無くなる。

５．使用する暗号技術への変更に問題がある。

新しい技術を使用するとき、その解読方法がすでに存在するならば暗号通信がむいみなものになる。

その有効性を利用者にきちんと証明できるのでしょうか？長い間、バックドアの存在を疑われていた暗号アルゴリズムも存在した。証明付きで安全と言われていた方法が解読されてしまった例もある。

古くなった暗号ソフトを、高い金を払って更新して、３年も経たないうちに解読だれたら誰が責任を取るのであろうか？

６．国による法律の違いに細かな対応が出来ない。

国や組織によっては、暗号メールを受け取れるが、暗号メールを送信できないような規制があったり、登録された暗号化鍵や復号化鍵しか使えなかったり、弱い暗号化しかつかえなかったりする。

暗号通信網に参加する人がそれぞれの通信相手ごとに、きめ細かく暗号化の設定や暗号化をしないような設定が可能でなくてはならない。IP アドレスから IP アドレスへの対応では、このことは不可能である。

以上の問題点は、私が提案する方法で解決できる。

１．パソコンから出るときに、データは暗号化されている。また、暗号化されたデータはパソコンの中で復号化される。したがって、暗号化されないデータが流れる通信経路は存在しない。

２．沢山のデータが異なるアルゴリズムを使って暗号化される。

私が提案する方法では、経済的、技術的な条件が許せば、それぞれの送信者のメールアドレスと受信者のメールアドレスの組に対して異なる暗号化アルゴリズムを使用することができる。

しかも、送信のときの暗号化アルゴリズムと、受信したデータが暗号化されているアルゴリズムは異なるものであってもかまわない。

したがって、同一の方法で暗号化されたデータの量は極めて少ない。

３．盗聴者は自分の作ったデータをその方法で暗号化できない。

最近のパソコンには起動時に生体認証を必要とするものがあり、ハードディスクの中のデータは暗号化されていることも普通のことである。

盗聴者が、通信経路に対象とする方法で暗号化された様々なデータを流してみるのは不可能である。

４．使用する暗号技術の決定が個人の判断でできる。

新しい暗号技術が開発されたとき、各個人がその内容を調べて信頼できる優れた技術ならすぐに採用すれ

ばよい。不安なら自作すればよい。責任は、自己責任である。
　各個人が優れた暗号ソフトを購入するなら、開発者の生活も保障され、さらに新しい暗号を開発できる。
５．使用する暗号技術への変更が自由にできる。
　信頼できない暗号技術はすぐに捨てて、新しいものに自由に変更できる。
６．国による法律の違いに細かな対応が出来る。
　国や組織によっては、暗号メールを受け取れるが、暗号メールを送信できないような規制があったり、登録された暗号化鍵や復号化鍵しか使えなかったり、弱い暗号化しかつかえなかったりする。
　暗号通信網に参加する人がそれぞれの通信相手ごとに、きめ細かく暗号化の設定や暗号化をしないような設定が可能でなくてはならない。
　メールアドレスごとに、送信の場合と、受信の場合で別々に設定できるから、通信相手の住んでいる国の法律に合った方法で暗号通信を利用できる。
　さらに、鍵を国に提出する場合でも、暗号通信網全体としては、様々なアルゴリズム、様々な鍵が存在するので、提出されるのはその中の僅かのものであり、その鍵が外部に流出しても、国際的な暗号通信網全体としての安全性は保たれる。

　以上の事柄は、暗号通信の中で大きな進歩であり、特許として認定されるべきであると考えます。これらが可能となったのは、拡張されたアドレス帳が、名前、アドレス、暗号化ソフト、暗号化鍵、復号化ソフト、復号化鍵の項目を持っているからである。
　これにあわせて、暗号化部分と復号化部分に分離された暗号ソフトを使うことによって強固な暗号通信網を構築できる。

Confirmation of the problems in the cipher network..
1. The length of the route of the flowing plane data is not minimum length.
　He assuming the existence of safety network and not safety network. Encryption is done at the boundary of these networks.
Where is safety network? All most of the leaks of important information from a organization is inside whistle blowing. The assumption of safety network must be throw away.

2. Too many data are enciphered using same algorithm.
　He decide the algorithm according to the pair of IP addresses. In the case of e-mail mail server has IP address. Many e-mail addresses using same IP address. As the result of this, too many data are encrypted using same algorithm.

3. Wiretapping person can flow his special data to this network.
　For decoding ciphered data, wiretapper collect many data. It is clear, if wiretaper can flow his special data to this cipher network, he will get much more advantage.
The provider of Internet can not distinguish whether his customer is wiretapper or not.

4. How decide the algorithm to be used?
　It is difficult to judge the existence of the backdoor of the cipher algorithm. The algorithm, which is selected by the caretaker of the mail server, may be not safety.
Who can accept responsibility? In the case of submission of keys to the government, can we continue this cipher network.

5. How change the algorithm.
　There is many cipher programs, which became to trash. If a new cipher program is adopted, and it is immediately decoded. Who should pay the cost of change the crypt software. Who can probe the safety of new software.

6. The problem of corresponding to the law of each government.
　The restrictions in each country are very different. Receiving the ciphered data is possible, but sending ciphered data is not allowed. The key must be registered. The length of the key must be short.
The agreement on the software, which should be used, will not reach to a point. What algorithm's program should be used, what key should be used.

My extended address book gives us the solutions for these problems.
1. When the data is flow out from a computer, it is enciphered. After the enciphered data reached a

computer, the data is decrypted. There is not exists the route of flowing plane data.

2. Many algorithms are used for enciphering data.
My extended address book gives us the possibility, different crypt algorithm can be used to each address. And the two algorithms may be different, from A to B and from B to A.

3. Wiretapper can not encrypt his special data using other person's cipher program.
Each person can use different algorithm. There is not exist common cipher program. Wiretapper can not know the algorithms of other person.

4. Each person can decide to use or not use of a new cipher program.
When new algorithm is given, each person can test it and decide the usage of it, on the responsibility of him self.
Each person pays the price of the program. The developer of cipher program can get the profit. This is the foundation of new and powerful program.

5. Change is easy.
If the method of decoding is appear, user can change the cipher algorithm.

6. Correspondence to the law of each country.
Each person should consider the condition of his partner, who is living another country.
Whether he can receive ciphered data or not. Whether he can send a ciphered data or not. For these conditions must be reflected on to the extended address book.

The columns of address, encrypting program, encrypting key, decrypting program and deciphering key, give us the settlement of these problems.
The extended address book is the opposition of database of cipher programs.

２７．答えは２２と同じですが、繰り返して記述します。
これについては、審査官の言われるように、バイナリーデータを電子メールで送るには、添付ファイルとして扱い、base64 などの適当な変換をして送信するのは昔からの技術です。
これを記述したのは、暗号化したデータはもとのデータと性質が異なるので通信網の既約に合致するようにしなくてはならない事を示したかったからです。電話などでは、あまり明確になっていないので、その必要性を強調したかったのです。
データは通信網のプロトコルに適合しなければ送信不可能なのは自明のことなので、削除すべき項目かもしれません。

As examiner says, it is clear that the treatment of the data must fit to the protocol of the network. This may be the description of natural things.

２８．Leppek の方法では、暗号化のためのデータベースが存在し、その中から使用する暗号化のための操作を選び出し、それを適用する順序を決定するために１６０が生成されて、使用されます。
この方法では、暗号通信網に参加するすべての人が共通の暗号データベースを持っていなくてはならないことになります。
新しい参加者に誰が、どのようにして共通のデータベースを配布するのだろうか？国によっては暗号の輸出や輸入には強い制限があり、暗号データベースを受け取った人がそれだけで犯罪者になってしまったり、送ったほうの人が犯罪者になってしまったりします。
極端に弱い暗号だけを集めたデータベースならば世界中に配布できる可能性はありますが、これでは使い物になりません。
データベースから使用する暗号操作を引き出すための検索テーブル自体がさらに大きな問題を含んでいます。それは、使用する暗号化鍵の部分が存在しないので暗号化ソフトと暗号化鍵を併せ持ったものを検索テーブルに登録していると考えられます。

安全な鍵の長さはますます長くなります。鍵の長さが４０００ビット程度になるとテーブルがハードディスクからあふれ出ます。
全体に共通な暗号データベースは使用しないほうが賢明です。

Leppek's system uses database for ciphering. In this system, every member of this network must have same database. If a new member is living in a foreign country, he will have many troubles for get the database.
The export of the database is not so easy. The new member may become an offender, when he received the database. If the database consist of weak cipher programs, the new user may be able to accept and use it.
Next problem is the size of table. This table does not having the column of key. The program includes the key, the database of this type's program becomes too big for storing. If the length of key is about 4000 bit, the size of reference table is too long.
We should not use the common database. If the common database does not exist, the user must verify the encryption program and decryption program independently.

２９．２４と同じですが繰り返して記述します。
たしかに、審査官の言われるように、暗号化の順序と、復号化の順序は逆でなくてはならないことは、自明です。
Schier は、
For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.
と書いています。
1. A encrypting the data to be sent to B.
2. B decrypting the data from A.
3. B encrypting the data to be sent to A.
4. A decrypting the data from B.
Schier は 1 と 3 を同じように考えています。これは暗号通信の対称性を仮定していることになります。
しかし、A さんが B さんに多重暗号化されたデータを送信するときに、使用するいくつかの暗号アルゴリズムと、A さんが B さんから送られてきたデータを復号化するときに使う暗号アルゴリズムは全く別のものを選べることが必要であり、それを実現するための拡張されたアドレス帳の構造は決して自明ではありません。
このような、拡張されたアドレス帳（多重暗号化の場合）のなかで、暗号化の部分と、復号化の部分に関してそれぞれ独立に適用する順序を変更できるところが新しい部分となります。

It is clear that the order of encryption and decryption must be inverse order.
We must consider four situations. With communication of A and B.
1. A encrypting the data to be sent to B.
2. B decrypting the data from A.
3. B encrypting the data to be sent to A.
4. A decrypting the data from B.
The preceding literature explaining case 1 and 2. But we must consider the relation of 1 and 3.

Schier says,
For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.

This means 1 and 3 are same, and he is thinking symmetrical cipher communication.

This is very difficult, if A can send ciphered data, but B can not send ciphered data.
Symmetrical situation is not natural condition.
In my method 1 and 3 are independent. In my extended address book, for same e-mail address, the program for encryption and decryption are independent. It is possible to use DES for encryption and to use AES for decryption at device 14.
The case 1 and 4 must be considered. In case 1 use DES algorithm and case 4 use AES algorithm. This is most different point of my method to the method of Schier.

As the result of this, we can change the orders of encryption programs, and change the order of decryption programs independently, in the case of multiple encryption.

３０．審査官の言われるように、復号化したものを音声に変えなくては電話になりません。
違いは、拡張されたアドレス帳を使って、A さんが B さんに"How are you?"と呼びかけるときの暗号化方式と B さんが A さんに"Fine thank you."と答えるときの暗号化の方式を別々のものにしたほうがより安全になると考えているところにあります。

As examiner says, it is clear the decrypted phone data must be changed to sound. The most important point is to use different encryption algorithms of A said to B "How are you?" and B answer to A "Fine thank you".
This is the occurrence of extended address book.

結論：私が提案する方法は、暗号通信網の持つ様々な問題点を解決し強固な国際的な暗号通信網を構築する技術を示したものであると信じています。

Conclusion:
The extended address book is the key of safety and powerful cipher network.
This system is the foundation of international ciphered network.
My method can solve many problems in the old cipher network.

今回は、審査官よりもしつこく書く事を心がけました。英語は変な英語ですが、私にとってはこれで精一杯です。
これをまとめて普通の英語にする弁理士さんは大変ですね。すみません。

８．２　反論提出（修正済み）（２００６年１０月７日）

弁理士さんは以下のように反論を修正しました。

Proposed response to the Office Action

Protected Communication System
Your Ref.: 058856-0109
Our Ref.: 01116UYMUS

1. Eextended address book of the present invention

First, there is an item of the property of the extended address book, i.e., the address, encryption software, an enciphering key, decryption software, and a decryption key, and an encryption portion and a decryption portion can be registered in the form where it became independent. Such table structure solves various problems in cryptocommunication.

By the pointed-out literature (cited reference), the symmetrical cryptocommunication using the database of encryption software is considered in many cases. This database and communicative symmetric property cause various problems. It is shown clearly [saying why a problem arises and] how asymmetric cryptocommunication and the extended address book solve these problems.

Example 1.

There is important information, including a customer's personal information etc., which cannot be taken out outside in a company. If the significance of the information from a customer is high, it will encipher and the information from a customer will reach a company. It is not realistic that the management receives and processes all code mails from a customer. After personnel receive and process the mail enciphered from the customer, it is realistic to report a required point to the management.

In order to prevent personnel carrying out customer data in the company to external, it forbids transmitting the information as which personnel were enciphered to external, and that code mail can be transmitted to external can consider the case where it is made only the management.

In this case, an office worker's mailer must be able to be set up not encipher the information to transmit although the received code mail must be able to be decrypted.

What is necessary is to separate an encryption portion and a decryption portion and just to treat independently.

Since the code mail from personnel is not sent to the mailer of the visitor who transmits code mail to personnel, encryption software and an enciphering key are registered into personnel's line (row) of a visitor's address book, but the column of decryption software and a decryption key is a blank.

Although the column of encryption software and an enciphering key of the line about this visitor of personnel's address book is a blank, decryption software and a decryption key are registered.

Example 2.

The case of the enciphered communication networks (telephone etc.) is considered. The function of the telephone and computer which can be used improves rapidly within a short period of time.

New machinery has a powerful calculation function. However, even if new telephone and a new computer are put on the market, some users of cryptocommunication can purchase it immediately. There are some users who will continue using the same machinery in about ten years.

What can be used by the telephone in which count ability is inferior, or computer is the easy encryption which does not need complicated calculation.

Suppose that Mr. A, Mr. B, and Mr. C are the members of a cryptocommunication network. Although Mr. A and Mr. B have the newest machinery, Mr. C assumes that old machinery is used.

Although Mr. A communicates with Mr. B and Mr. C, he presupposes that Mr. B is communicated only with people with new machinery.

In order for Mr. A and Mr. C to perform cryptocommunication smoothly, communication is smoothly impossible unless it adopts the easy cipher system suitable for the count ability of the machinery which Mr. C has. It is because a decryption will take time too much if powerful encryption is chosen. Since time is taken too much after receiving the signal enciphered by Mr. C's machine until it becomes voice and is audible, it stops functioning as a telephone.

A code database becomes many things useless for people who do not perform communication with those who have old machinery among people with the newest machinery. It

is a waste of memory to save the old software which is not used.

Since memory can utilize effectively supposing it performs cryptocommunication only by those who have machinery of the latest style except for Mr. C, more powerful encryption can be used. However, a communication network will be destroyed whenever new machinery comes out, if Mr. C is excluded.

All the members should give up the method using common encryption software, and each user should manage encryption software uniquely. It should be made to use what suited Mr. C's Mr. C in what suited Mr. A's Mr. A.

The following is done to achieve more powerful communicationby by encryption.
If both of them has a machine of high performance, the powerful code can be used, and one of ot the other has an old machine, only an easy code should be used.

Therefore, Mr. A's address book is as follows:

The encryption software of Mr. B's line is powerful, and an enciphering key will become long. Of course, the decryption software of Mr. B's line is powerful, and a decryption key will become long. Of course, the software which uses another cryptographic algorithm may be sufficient as encryption software and decryption software.

What has an enciphering key easy the encryption software of Mr. C's line and short is registered. Of course, the decryption software of Mr. C's line is easy, and a decryption key will become short. Of course, the software which uses another cryptographic algorithm may be sufficient as encryption software and decryption software.

Mr. C has to register only easy encryption software into his own machinery altogether according to old his own machinery.

Therefore, Mr. C's address book is as follows:

Since Mr. B and Mr. C do not communicate, Mr. B's line does not exist.
What has an easy encryption software of Mr. A's line and short enciphering key is registered.
What has an enciphering key easy the encryption software of Mr. A's line and short is registered. Of course, the decryption software of Mr. A's line is easy, and a decryption key will become short. Of course, the software which uses another cryptographic algorithm may be sufficient as encryption software and decryption software.

Since Mr. B's communications partners are only people with the newest powerful machinery, all the encryption software registered into Mr. B's machinery is made into a powerful thing. It is not necessary to put the old software which is not used on its own computer.

Therefore, the extended address books differ in Mr. A's, Mr.B's, and Mr.C's respectively, and it is obvious they are unrelated to a common database.

[the way of thinking which builds the database of encryption software and unifies encryption software on the whole]

When renewal of a database is difficult, if old encryption software is discarded with whether it will be kept outdated and updating according to the count ability of machinery with the old whole, one of the methods whether to banish people with old machinery from a cryptocommunication network must be chosen. Though added, if it does not discard, unnecessary encryption software will be kept in large quantities, and memory will be used vainly.

Of course, it becomes clear from Example 3 that renewal of a database and unification are the most difficult subjects. However, since the structure which the encryption column and the decryption column can set up freely for every communications partner exists there if the address book in which the place which threw away unification by a code database and the applicant proposed was extended is used Powerful encryption software is registered to people with highly efficient machinery, and since easy encryption software can be set up to people with old machinery, while a new thing and an old thing fully employ the function efficiently, it can live together in a cryptocommunication network.

Example 3.

There is a still more difficult problem in a code database. The 1st is the limit about the activity of a code.

There is a big difference in the policy about cryptocommunication by a country. The country which is not allowed to transmit although it is possible to receive the country which can perform communication enciphered freely, and the enciphered data, and its reverse country. Although the country allowed neither transmission of the enciphered data nor reception and the activity of a code are permitted, their country where a limit is attached to the treatment of an algorithm, an enciphering key, and a decryption key is various.

It is the time when internationalization is cried for. It is common that communication crosses the border. About a foreign friend, communalization of encryption software becomes precocious only to an offender.

For example, as long as mechanical capacity allows, cryptocommunication should be carried out to the friend who lives in the country which uses the enciphering key of the length in the limit to those who live in the country where a limit is attached to the length of a key, especially does not have a limit using the long key for making a powerful code.

It will be severer when those who live in the country which asks for presentation of an enciphering key and a decryption key participate in this cryptocommunication network. It is because the person of the position which can decrypt all the codes registered into the code database will exist when the information on the accumulated key flows out.

Don't see from this point or don't unify the whole. Though the law of each country is protected, in order to realize the safest possible communication, everybody need to manage their own address book freely, and register the optimal encryption software. And only the part of those who do cryptocommunication directly with themselves is managed. Even if cracking of a certain person's computer is carried out, there is little influence on the whole cryptocommunication network. People who perform direct communication with the person by whom cracking was done should transpose only people's line by which cracking was carried out to new software and a new key.

If it is the applicant's method, the influence by presentation of a key is also slight. [since only the key used for communication with Mr. B will be submitted to Mr. B's country through Mr. B if Mr. B who is one person of the communications partner of Mr. A who lives in a country without regulation lives in the country upon which a duty of presentation of a key is imposed, only Mr. B's part is influenced, but] It is because the secret of other portions and the portion used for communication with those who live in the country which does not demand presentation can be maintained.

Safety becomes firm just because information is not accumulated. The person who obtained a new computer can update immediately. It is possible to lengthen the length of a key at least.

The 2nd is a trouble of export.

A still stronger limit is attached about exporting encryption software. When a code database is common, the encryption software used as an expert embargo is contained in it. Who distributes the copy of a database?

Even if it is going to distribute as free software in the form downloaded from the Internet, there is also a country where it is considered the act which exports powerful encryption software to a foreign country, and the distribution person of encryption software turns into an offender.

He will be an offender just as it builds the cryptocommunication network using this code database. If only the easy thing that export is permitted is extracted and distributed from a code database, similarity is lost and it stops functioning as a code database. The code database cannot build an international cryptocommunication network.

By the method of using the extended address book, it corresponds as follows to this trouble. A firm code can be used between a country without legal regulation, and a country without legal regulation. What is necessary is just to choose the most powerful thing within regulation, when there is regulation. Probably, it will be common for the regulation in the case of receiving code data to be weak. Because, it is since many persons will be made into an

offender by the enciphered spam mail. If the regulation when transmitting and the regulation when receiving have a difference, the firmest thing can be chosen in the form suitable for it.

The 3rd is a problem of copyright.

Sharing of encryption software may disregard the copyright of the manufacturer of encryption software. Who pays the payment of the fee when distributing a copy how?

When adding a new code to this code database, that copy must be distributed all over the world. Do all the members assent to pay an owner of a copyright money?

The person who refused payment will be excluded from this cryptocommunication network.

If all the members share a copy, the activity of encryption software other than what was specified becomes impossible. The encryption software made into a standard is changing from DES quickly like Triple-DES and also AES. The length of the key made safe [RSA cryptograph] becomes long rapidly with 512 bits, 1024 bits, and 2048 bits.

When throwing away the cipher system which became old and adopting a new cipher system, it may become one of whether the illegal copy of it is carried out, or all the members purchase. Or even if it is going to distribute as free software in the form downloaded from the Internet, there is also a country where it is considered the act which exports powerful encryption software to a foreign country, and the distribution person of encryption software turns into an offender. There is also a country which says domestic that distribution is possible by mail. Distributing new software by mail also requires costs. Who pays a postage?

In order to use powerful new encryption software, the method of guaranteeing a developer's proper profit is required. This will become possible if it is the extended address book which the applicant have proposed, and the encryption software suitable for it.

If a new code is developed, a required person pays money and can purchase immediately. It can be used immediately, and considering the mechanism in which a developer's profit is moreover guaranteed, separation of encryption software is needed. It is separation of an encryption portion and a decryption portion. And redistribution of prohibition of redistribution of a decryption portion and an encryption portion is free. Creation and distribution of a key must be possible.

In order to correspond to this separated encryption software, it is required for the column of encryption software, the column of an enciphering key, the column of decryption software, and the column of a decryption key to be independently.

RAS of a long key needs a powerful calculation function. It cannot respond by an old machine. If priority is given to the encryption software share which even a key includes, the old machinery which exists in a communication network also here will threaten the safety of cryptocommunication.

2. General comment to the cited references [Regarding Vidrascu]

col. 6 line21-31

When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.

The transmission is next sent to the interface 31; When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30.

It came to be able to use the mail of "user@domain" form in 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "user12@domain1" have the same mail server which having a same IP address.

Moreover, in the communication from "user11@domain1" to "user21@domain2" and also from "user12@domain1" to "user22@domain2", the same encryption will be done.

It came to be able to use the mail of"User@domein" form in 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "HYPERLINK mailto:user@domein" have the same mail server which having a same IP address.

Moreover, in the communication from "user11@domain1" to "HYPERLINK mailto:user@domein" and also from "HYPERLINK mailto:user@domein" to "HYPERLINK mailto:user@domein", the encription will be done.

There are very many people using the same provider, and the users using the same IP address increase in number inevitably. Therefore, with the same method and the same enciphering key, a lot of data will be enciphered and it will transmit. This becomes a big cause by which a code will be decoded.

The response between IP addresses means the response between mail servers. Now, it will be only a manager of a mail server that encryption software can be changed. There are much people using the same provider's mail server. The group of the people using two mail servers of them will use the same encryption software. Now, each user of an E-mail cannot change encryption software freely. It must be able to change, immediately after decode art is found.

In order for the user of an E-mail to choose the encryption software which suited itself and to use freely, encryption software and an enciphering key must be able to be set up not according to the response of an IP address and an IP address but according to the response between mail addresses. If he purchases new machinery, it should enable it to change into a firmer code. If this is not made, cryptocommunication with those who live in the foreign country where legal limits differ will be the origin of a trouble. For realizing people of various countries registered into their own address book, and smooth cryptocommunication, the user of an E-mail has to be able to set up finely freely in his own address book.

It is not written that choosing the key of a code to compensate for an IP address sets and chooses encryption software as an IP address although written. The encryption software which appears in this is only DES, and description which uses the code of an algorithm which it is at the transmission [to the same partner] and reception time from the same partner, and is different does not exist.

This method includes the further trouble. It is the point of dividing and considering a safe network and the network which is not safe, and trying to perform encryption and a decryption on the boundary. The problem is whether the local network made safe exists. The customer data of a company are sold and bought at a high price. A possibility of intercepting the packet which is flowing without enciphering personnel in the local network made safe, and selling outside is also fully considered. Most leaks of information are the forms where human being in the company makes confidential information flow into external. Therefore, the boundary of a safe network and the network which is not safe etc. does not exist. It needs to be enciphered when coming out of its own personal computer. Therefore, management of encryption software must be directly carried out with each personal computer, and everybody need to have the address book extended into its own personal computer.

col.12　line 1-11

if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.

verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and if the first and second keys exist, deciphering a received message with the second key and otherwise reject a received message.

Who can check how where the 1st key and 2nd key exist poses a problem immediately.

The 1st key is used for the encryption at the time of transmission, and the 2nd key is used for the decryption in the side which receives the enciphered data. Suppose that Mr. A sends the data enciphered to Mr. B.

Even if Mr. A checks that the 1st key and 2nd key are in their place, it does not mean having checked that the 2nd key was in Mr. B's place. Even if Mr. B is telephoned and it checks before Mr. A's transmitting, judgment whether Mr. B is really checking or the lie is told cannot be performed. Mr. A has to go to Mr. B's place before transmission, and he has to check existence of the 2nd key by himself.

In international communication, it is not used.

First of all, it is not necessary to investigate the 2nd key that cannot be checked. It is enough, if it will encipher and send if there is the 1st key then. If it does not arrive well, there is a notice from Mr. B.

Checking here is not checking existence of encryption software only in existence of a key. It is an item which must be checked if it may not encipher as the case where use many kinds of encryption software, or it enciphers. It is because it does not consider saying that encryption software considers only DES and the thing without the description will change the kind of encryption software in the time of transmission and reception (for example, DES) (for example, AES) at all.

If change of encryption software is considered, the description about whether what we do with change and registration of encryption software by who determining in what kind of position will be required.

Four must be considered if you consider a key.

1. Key used when enciphering data which Mr. A sends to Mr. B.
2. Key Used when Decrypting Data Which Mr. B Received from Mr. A
3. Key Used when Enciphering Data Which Mr. B Sends to Mr. A
4. Key used when decrypting data which Mr. A received from Mr. B.

Encryption software is also the same.

1. Software used when enciphering data which Mr. A sends to Mr. B.
2. Software used when decrypting data which Mr. B received from Mr. A.
3. Software used when enciphering data which Mr. B sends to Mr. A.
4. Software used when decrypting data which Mr. A received from Mr. B.

By the applicant’s method, it does not carry out checking existence of a certain thing to a remote place. It is because it does not become the certification of existing now even if suited in the past.

It is an enciphering key and the encryption software which are checked at the time of transmission. Because those who return only the data which is not enciphered are also needed even if it receives code data.

Naturally at the time of reception, only existence of a decryption key and decryption software is considered. It is because the data enciphered to the person may be unable to be transmitted even if the enciphered data is receivable from the person. The existence of a country by which offender treatment will be carried out only by receiving the enciphered data is also considered.

Although code mail is unreceivable, it thinks, also when code mail can be transmitted. Therefore, if only an enciphering key and encryption software are not investigated at the time of transmission and it is not made not to investigate only a decryption key and decryption software at the time of reception, the cryptocommunication beyond the border becomes impossible (even if it can encipher only one of the two).

By this literature, when it decides to use the cipher program of a public key system, a big problem arises. In a public key system, the 2nd key is a secret key and it is not carried out to

a sending person with Lycium chinense. This method cannot be equivalent to a public key system. What cannot respond to a public key system is already outdated.

[Regarding Schier]

col.9 line67, col.10 line1-7
The receiving device 78 includes a copy of table 80. The copy of table 80 is then used at step 132 to retrieve the first indicated encryption algorithm from the encryption algorithm column 84. This encryption algorithm is then loaded into the encryption decryption engine 30 and is executed by central processing unit 20 to encrypt outgoing communication and decrypt incoming communication at step 134.

This trouble is having a copy of table 80. It is that the decryption software specified by the same number as the encryption software specified by the number of the encryption algorithm indicated on the table exists in both 78 and 72.

Mr. B and Mr. C will also have the same table, this will become ... etc., if Mr. A and Mr. B have the same table, and all the members will have the same table.
It contains a problem in some respects that those [all] that participate in a cryptocommunication network have a common table.

1. Can this Table Perform Addition, Deletion, and Updating?
2. Who Makes Decision of Addition, Deletion, and Updating?
3. How is Added Software Distributed?
4. Who Pays Royalty how?
5. Doesn't Distribution beyond Country Become Criminal Act?
6. When a table changes to a powerful new code, can the owner of old machinery use?
7. Why Do People with New Machinery Have the Need of Saving Old Code Which Cannot be Used?

If a table is shared, such a problem will be able to be thrust before. Sharing of a table must be given up. On the contrary, never sharing is a basic principle which builds a firm cryptocommunication network.

It solves the above-mentioned conflict that only its part manages the address book in which everybody were extended.

Using a copy will share the same encryption software as the table with all the same machinery that participates in this cryptocommunication network, and the same decryption software.

Since this encryption communication network accepts intervention of outdated machinery, the weak encryption which is used also by low count ability will be used.

Since all the machinery has a copy, if the number of the machinery which participates in the encryption communication network increases, changing into new cryptographic algorithm will become very difficult. Because, it is because there are those who can pay the royalty of encryption software or decryption software, and those who cannot pay, and the law of a different country may allow the activity of the algorithm, or may not be allowed. There are some countries which have forbidden carrying out powerful encryption software abroad, and there are also some countries which have allowed carrying out freely. If a copy is distributed on the Internet, it will become a criminal act in many cases.

Although a problem is not used in a certain country about the data encryption to receive, it thinks, also when applying a limit to the data encryption which transmits.

For instance, there is such a case. The code can be freely used in the country where Mr. A lives. However, in the country Mr. B lives, when receiving, the code can be freely used as well as Mr. A, although there is a limitation which only an easy can be permitted on transmitting.

So, in the country of Mr. B, the encryption software should be easy and with a short enciphering key.

If the table and encryption software of encryption of Mr. A are sent to Mr. B, he will be an offender only by Mr. B using it. It is actual that legal limits differ about a code. In order to build as effective a cryptocommunication network as possible in this, the extended address book (telephone number book) which everybody manage uniquely is required.

There is also a country which is asking for submitting a decryption key to the public institution of a country. If the copy of a code database is used, the country which submitted the key can decode all the communications. Since the information on such an accumulated decryption key and an enciphering key can be sold at a high price, worries about the leak of information from the interior of a public institution also take place. If those [all] that participate in a network share encryption and a decryption, it will have the capacity which all the members decode.

For the above reason, this method is not fit for internationalized communication. A copy must be given up. However, this difficulty is solvable if the idea of an address book that the applicant was extended is used.

[if encryption software is only called from the telephone number, are an easy talk, but]

Encryption software is divided into an encryption portion and a decryption portion, and it is sure that it is the most important point to extend an address book in the form suitable for it, and to have enabled it to register encryption software, an enciphering key, decryption software, and a decryption key. This point is not described at all. Never don't perform description which distributes a copy so that it may not destroy a cryptocommunication network itself.

By the applicant's method

Mr. A and Mr. B buy new telephone, and suppose that Mr. C is continuing using old telephone. At this time, by the method which I propose, the encryption software and enciphering key which are registered into the line of Mr. B of the address book which Mr. A has can register the newest thing, and can use it.

Since Mr. C has only old telephone, also when performing cryptocommunication with Mr. A, and also when performing cryptocommunication with Mr. B, he uses an easy cipher system. Since a decryption and encryption take time too much by old telephone, a firm new code cannot be used. Therefore, old, easy encryption software must be registered into the line of Mr. B of the address book in which Mr. C was extended.

Therefore, at all, the address book in which Mr. A was extended, and the address book in which Mr. C was extended are another, and are not a copy.

The way of intensity as the whole cryptocommunication network where the individual can be made to do a setup of encryption and a decryption according to the performance of the telephone which he has, and the performance of the telephone which its communications partner has, respectively improves.

Even if it sees from this point, five items, the address, encryption software, an enciphering key, decryption software, and a decryption key, are needed for the item of an address book or a telephone number book.

Each one of signal books are managed to everybody. Communalization is not carried out. This is important. If it carries out like this, new machinery can make the environment where the force can be demonstrated immediately. Old machinery must not threaten the safety of a cryptocommunication network. It is important that the contents of the address book extended in the form which suited everybody's situation of its communications partner can be adjusted.

[oneself and]

Even if it sees from this point, the extended address book improves the safety of cryptocommunication by leaps and bounds, and it is the method of moreover also solving the legal trouble in international cryptocommunication, and is sure that prior art is excelled.

It is because it thought that that this became possible would encipher by respectively separate encryption algorithm by dividing into two, the communication of information which had the directivity from Mr. A to Mr. B for the conversation of Mr. A and Mr. B, and communication of information with the directivity from Mr. B to Mr. A.

I hear that the conversation {A, B} of A and B was decomposed into two, (A->B) and (B->A), and separate encryption was introduced to each when symbolizing.

[Regarding Leppek]

col.2 lines 19-23, 51-55, col.4 lines 14-17
The fundamental machine of the virtual encryption scheme of the involves the generation of a sequence of the access code, with immediately successive ones of the access code of the sequence being different from one another.

col.2 lines 51-55,
The order of the encryptors within the sequence to which the data is applied may very as desired, and the sequence may 'toggle' or switch back and forth between the same set of encryption operators as part of its overall encryption flow.

col.4 lines 14-17
The encryption routines 110, in and of themselves, need not be any particular type of encryption algorithm and may be conventional encryption operations, such as PGP, DES, etc. routines, as non-limiting examples.

The phenomenon which should consider only the scene where multiplex encryption is carried out and data is transmitted, and the scene decrypted after arriving at an addressee's place, and was alike happens.

There are many troubles in the method of this literature, and it is not realistic.
1. Is Database Common to All the Members?
If common, how is the problem of copyright and an illegal copy solved?
How do an economic discrepancy and a legal wall exceed?
If not common, how does the algorithm which makes a number from a key correspond to this unbalance?

2. Isn't there any limit in table size?
When changing the length of an enciphering key with the same algorithm, what does it do?
In key RSA which becomes long rapidly, it seems that it will be about 8000 bits in the near future although it is the present 2048-bit key.
It becomes impossible for the length of the table of a database to become the 8000th power of 2, and to manage the whole.
The 8000th power of 2 is the 2400th power of about 10, in about 100 G bytes, this is 100,000,000,000 bytes and the hard disk of the present personal computer is only the 11th power of 10.
It is impossible it not only to be unable to to manage, but to manage this database in a table.

2. Is Soft New Addition Possible?
How does it tell having added?
When many persons add a new thing to random, respectively, what happens to the sequence of registration?
If a number is carried out, what is it made in common?
Or isn't it necessary to make it a common number?
How is the soft new number to be used decided?
How is the software made for it to be new and to use distributed?

3. Is Old Soft Deletion Possible?
Must a thing old till when be saved?
Can agreement of all the members for deleting be taken?
If it deletes, do you pack a number and pack it?

4. Is Rewriting Possible?

is it alike once, and do you delete and can register with the place newly?
When it can do, who is told about change of a number how?
How is the software to be used redistributed?

5. What Happens to Determination of Number after Change?
Does the algorithm which generates a number generate a different train to the same key?
If it deletes, do you generate the new train which excepted it?
If this adjustment is carried out, how is it possible?

6. What Has Happened to Algorithm of SUPERVISOPY ENCRYPTION MANAGER Which Determines Number?
Can a new code be added to a database?
When it adds, are they change of the train to generate and an algorithm which can be adjusted?
the same thing as the existing whole database can be added -- ? (a law, copyright)
How does it correspond to a new participant?
How can the number formation algorithm respond to a new participant?
Does the new participant have to register the software which is old and does not become useful?

7. What Does ENCR.KEY Use?
Is it the address?
How is the train of 160 generated to a new participant?
Was the number generated to a new participant already decided?
Or can a participant set up the train generated freely?
If it cannot set up, using an outdated code is forced?

8. Is the Same Train Generated if what who is Using Puts in the Same Key?
Does what the tapping person has teach the turn of a decryption similarly?
Or can't a user see the data?
Can't it be operated?
Is change impossible?

As seen above, even if it decides the turn enciphered from an addressee's address, it will not be clear how it deals with updating, deletion, new registration, etc. by the database structure having a trouble. It is not practical.

Conclusion,
1. Don't use the database which manages the whole.
2. Everybody must be made to have to manage only a part to use for one's communication.
3. Everybody manage the address, encryption software, an enciphering key, decryption software, and a decryption key for every communications partner.

Thus, my method based on its own extended address book is simple for registration and updating, and is the only method of moreover clearing the problem of an illegal copy, and a legal problem.

[Regarding Collins]

Col.5 lines 43-63
The Internet e-mail gateway 240 converts the SMS message 100 to an Internet e-mail message having the format 150. To accomplish this, the Internet e-mail gateway 240 determines an Internet e-mail address for the cellular telephone 210 and for the Internet station 280. Regarding the Internet e-mail address for the cellular telephone 210, the Internet e-mail gateway 240 adds the domain name for the Internet e-mail gateway 240 to the SMS address for the cellular telephone located in the sender SMS

address field 110 of the SMS message. To determine the Internet e-mail address for the Internet Station 280, the Internet e-mail gateway 240 uses a messaging service lookup table having a first column with a list of SMS addresses, where each SMS address corresponds to an Internet e-mail address in a second column of the lookup table. The Internet e-mail gateway 240 locates the SMS address of the Internet station 280 in the first column of the lookup table. The Internet e-mail address for the Internet station 280 is retrieved from the second column of the lookup table at a location corresponding to the Internet station SMS address of the first column.

Here, the response relation between the SMS address and an internet mail address is only described.

Now, the structure of the extended address book cannot be discovered.
Because, the method of enciphering signal transfer in case Mr. A calls to Mr. B in communication of Mr. A and Mr. B, It is because it is not solved why it must be made separate, although it is also possible to make the same the method of enciphering signal transfer in case Mr. B replies to Mr. A corresponding to it and it is easier.

Since this break through is insufficient, advanced technology includes various troubles and has made development of cryptocommunication difficult. Though development of cryptocommunication is desired, only a thing of structure which blocks it is offered.

It is difficult to notice the need. This is just a nucleus portion of the present invention of me. Though it corresponds to the difference between copyright and the law for every country, the difference in mechanical count ability, etc., for building a cryptocommunication network firm as a whole.

The mechanical performance provided with five items of the address, encryption software, an enciphering key, decryption software, and a decryption key in which he has [the individual] the extended address book, Just the method which considers only the thing of each individual's communications partner and can be developed freely is required, and various problems can be solved by this.

This idea became five items of need for an address book just because it thought that two different cipher systems were used for communication of Mr. A and Mr. B, and it led to separation of the encryption portion of software, and a decryption portion by it. The solving means of a copyright problem or an illegal copy problem has been offered by this. It is sure that this point is big progress.

Such a point of argument about the extended address book is not solved at all in the literature in which the examiner was shown. Therefore, I think that the applicant's method has suitable unobviousness in being patented.

[Regarding Keats]

Each network element has its own SID (maintained for example in a memory element under control of the SID manager) and also maintains a database, table or other suitable structure in another memory element, such as a cache, which maps TIDs to IP address for IP connections from the particular network element to another network element in the network, as described in detail below.

Changing data using table structure occurs from ancient times. An address book, a telephone directory, various mathematical tables, etc.
However, various troubles in cryptocommunication are what, and in order to solve them, nothing is described about whether the table of what kind of structure is required.

It must be said that the trouble mind which should be solved is not attached.

3. Opinions to the examiner's statement

[Regarding item 8]

“two separate storing area” is clearly described in the claim 1 currently amended as follows:

“the reference table having a plurality of rows each corresponding to a single electronic mail address, an addresss column storing an electronic mail address, an enciphering column which includes a key area storing an enciphering key identifier and a software area storing an enciphering software identifier, and a deciphering column which includes a key area storing a deciphering key identifier and a software area storing a deciphering software,”

Accordingly, we believe that the examiner’s assertion “claim language does not disclose separate table columns for deciphering and enciphering programs” is overcome.

[Regarding item 9]

As for a “one-on-one relationship”, the applicant withdraws his statement, that is, “a one-on-one relationship between the sender and the receiver, which creates the ordered pair cannot be achieved using art of record because the enciphered packet is designed to be recorded by more than one terminal.”

[Regarding item 13]

As for the phrase “application order data”, the applicant adds a qualifier to the phrase in claim 1 currently amended as follows:

“-------application order data indicating an order in which each of the pairs of the enciphering key and the enciphering software is applied-----”

[Regarding item 15]

As for “whether --------correspond separately to a main text and an attachment, or whether they-------separately for each email address.”, the applicant amends claim 3 and 4.

[Regarding item 16]

As for “The limitation is not clear.”, the applicant amends claim 6 as follows:

“means for deciphering and outputting voice of the received voice signal in order by using the deciphering program, wherein said deciphering program to be used being specified based on the data of the transmission source telephone number of the communication, and the deciphering reference data.”

[Regarding item 17]

When first the point of argument of Vidrascu, Leppek, Collins, and Schier is arranged and they are synthesized, I would like to describe whether it is inventing how far ordinarily.

The feature of Vidrascu has raised the cryptocommunication between IP address IP addresses. The communication network which is safe and not safe are divided and considered. The number of the cipher systems currently treated is one, and they are checking further existence of the key used for a decryption by a receiving side at the time of transmission.

The feature of Leppek serves as a method which uses the database treating multiplex encryption of encryption software. It is the algorithm indefiniteness which determines the encryption software to be used, and, as for something, the key used at this time is not specified.

The feature of Collins is treating the response of the SMS address internet mail address.

The feature of Schier shares the copy of table 80. Of course, the way of thinking which transposes that of the number of a key to the telephone number may be able to be thought of. However, now, they are only response profitable of the telephone number and encryption algorithm. Sharing a copy will have the same table wholly. Registering with this table are all

telephone number. Such a table is meaningless and is not realistic.

Now, probably, it will be possible to invent to make an internet mail address and the telephone number correspond, since Collins showed.
It is not difficult to transpose the IP address of Vidrascu to an internet mail address, either. The applicant do not think that an enciphering key is so difficult to treat.

If multiplex encryption of Leppek is used, making a conversion table will not be so much difficult so that the multiplex encryption of the data transmitted to the address can be carried out to an internet mail address.
The way of thinking which transposes that of the key number of the table of Schier to the telephone number may be able to be thought of. However, now, they are only response profitable of the telephone number and encryption algorithm. Sharing a copy will have the same table wholly. registering with this table storing all the telephone number. It is meaningless, and it comes out, and there is such a table and it is not realistic.

Therefore, even the conversion table of a mail address, encryption software, and an enciphering key thinks that it is a matter invented ordinarily. The ways of thinking which share a code database and to say are also Vidrascu and Schier.
Then, it is common.

Although the situation when transmitting encryption data to Mr. B and the situation when decrypting Mr. [B] which received it are indicated from Mr. A,
The description when processing what Mr. A received from Mr. B is missing. It is because it thinks that the reason this is missing is the same as the time of Mr. B treating what has been sent by Mr. A in an encryption database. That is, it is because cryptocommunication is considered as a symmetrical thing.

This is completely different. It is the difference in whether cryptocommunication is considered to be a symmetrical thing or an asymmetric thing is considered.

Only asymmetric cryptocommunication needs this break through. The break through about processing in case Mr. A receives from Mr. B just because it is asymmetric is needed.

Four considerations are required.
1. When Mr. A transmits to Mr. B.
2. When Mr. B processes data sent by Mr. A.
3. When Mr. B transmits to Mr. A.
4. When Mr. A processes data sent by Mr. B.

At least, these four points must be considered.

Consideration of only 1 and 2 is managed, and although the remainder sets symmetric property of the communication assumed tacitly as a tacit premise therefore, it serves as insufficient consideration.

For attaining the contrast of 1 and 4 performed by Mr. A's hand,
The consideration to address, encryption software, an enciphering key, decryption software, and a decryption key is indispensable.

This consideration draws the asymmetry of cryptocommunication and realizes destruction of a code database. This solves various problems.

However, the most difficult thing,

Let decryption software and a decryption key other than encryption software and an enciphering key correspond to a mail address. And it is never making a common code database.

If you can see signs that this response solves various problems, he can understand why it is important and the addition of the decryption portion which looks simple apparently will regard it as the difference in an awareness of the issues supporting it becoming clear. And if not

sharing a common code database checks important-ization why for international communication, it turns out that the problem which the slight difference on appearance solves is very large.

Something to solve and art have a close relation. I think it the same as whether the issue which should be solved is invented ordinarily whether it invents ordinarily. The applicant's method solves many troubles and it is sure that it is the method which is a way a powerful cryptocommunication network is realizable, and progressed.

Although it will see in order for details, the outline is seen simply here.
First, the trouble of Vidrascu is checked.

Vidrascu has raised the method of his own cryptocommunication. However, the method of Vidrascu is holding many troubles and does not give the method of very safe communication.

Troubles are enumerated below.
1. The path length into which the signal which is not enciphered flows is not the shortest.

In the network made safe, the data which is not enciphered flows and it is enciphered on a boundary line with the network made not safe. The place which assumes the existence of a network recognized safe is the biggest trouble. It is the feature of a present-day information crime which information reveals from human being inside an organization in many cases.

2. It is enciphered using an algorithm with much same data.

In the structure of an E-mail, many mail addresses correspond to the same IP address. Many people using one mail server are using the same IP address. Therefore, when a cryptocommunication network is built based on the response of an IP address and an IP address, many E-mails will be enciphered by the same method. This serves as a big key of decryption.

3. A tapping person can also encipher the data which he made by the common method.

If a tapping person registers himself as a user of the mail server made into the purpose, the data which is useful for decryption will be transmitted, the feature of an algorithm will be caught, and a possibility of succeeding in decode will become high.

4. There is a problem in the determination of the encoding technology to be used.

Judgment whether the encryption software which there is a backdoor and can be decoded easily is used is difficult. It is also considered enough that he cannot feel easy about the method which the server manager chose.
When new encoding technology is developed, how much do those who are going to encipher the data which manages a mail server and passes through that pay a developer a software price?

There are not so many mail servers. Only a small number of encryption software will be adopted. The developer's development costs will not fully be filled up.

Who has charge in the safety of a code, as for the further problem? There is also a place required as some country and organizations submitting an enciphering key and a decryption key. The meaning of cryptocommunication will be lost if this information leaks.

5. There is a problem in change to the encoding technology to be used.
When using new art, cryptocommunication will become meaningless if the decode method already exists.

Can the effectiveness be proved exactly to a user? The cryptographic algorithm whose existence of a backdoor was being doubted had also existed for a long time. There is also an example in which the method said to be safe with certification has been decoded.

If it decodes before paying high money, updating the encryption software which became old and three year's passing, who will take charge? Who will pay the costs when updating?

6. A fine response is not made to the difference in the law by a country.
Although code mail is receivable for some country and organizations, there can be no regulation which cannot transmit code mail, only the enciphering key and decryption key which were registered can be used, or only weak encryption is obstructed.

A setup that those who participate in a cryptocommunication network do not do a setup or encryption of encryption finely for every communications partner must be possible. In the response to an IP address from an IP address, this is impossible.

The above trouble is solvable by the method which the applicant propose.
1. Data is enciphered when coming out of a personal computer. The enciphered data is decrypted in a personal computer. Therefore, the communication path into which the data which is not enciphered flows does not exist.

2. It is enciphered using the algorithm with which much data differs.
By the method which I propose, if economical and technical conditions allow, different encryption algorithms to the group of each sending person's mail address and an addressee's mail address can be used.

And encryption algorithm in case Mr. A transmits to Mr. B may differ from the algorithm with which the data which Mr. A received from Mr. B is enciphered.
Therefore, there are very few amounts of the data enciphered by the same method.

3. The tapping person cannot encipher the data which he made by the method.

A tapping person is unable to pour various data enciphered by the communication path by the target method.

The method of encryption is unknown. If it understands, there will be no need of pouring and testing data.

4. Determination of the encoding technology to be used can be performed in judgment of an individual.

What is necessary is just to adopt immediately, if it is the outstanding art with which an individual can investigate and trust the contents when new encoding technology is developed. What is necessary is just to make on one's own, if uneasy. Charge is self-responsibility.

If the encryption software excellent in the individual is purchased, a life of a developer is also secured and a still newer code can be developed.

5. Change to the encoding technology to be used can be performed freely.
The encoding technology which is not reliable is thrown away immediately and can be freely changed into a new thing.

6. A fine response is made to the difference in the law by a country.
Although code mail is receivable for some country and organizations, there can be no regulation which cannot transmit code mail, only the enciphering key and decryption key which were registered can be used, or only weak encryption is obstructed.

A setup that those who participate in a cryptocommunication network do not do a setup or encryption of encryption finely for every communications partner must be possible.

Since it can set up independently for every mail address the case of transmission, and in the case of reception, cryptocommunication can be used by the method suitable for the law of the country in which the communications partner lives.

These became possible because everybody managed uniquely that the extended address book has an item of the address, encryption software, an enciphering key, decryption software, and a decryption key, and the extended address book.

A firm cryptocommunication network can be built by using the encryption software divided into the encryption portion and the decryption portion in accordance with this. It is

important to never have a common code database.

It becomes clear [in the counterargument to the following items] for details.

[Regarding item 18]

[that One Person Has] if RSA and ECC are Used by Public Key Cryptosystem when it Thinks of Two Persons Who Communicate by Method Which I Propose It is the decryption software of RSA, the secret key of RSA and the encryption software of ECC, and the public key of ECC that the encryption software of RSA, the public key of RSA and the decryption software of ECC, the secret key of ECC, and a partner's person have, and a key will be set to four if two persons' thing is totaled.

Of course, there is no need that these four keys exist simultaneously, and only the part may exist. When transmitting the enciphered data, it is enough just to check that there are encryption software used as a sending person and an enciphering key. Whether it has a key for a decryption of an addressee does not have the need for a check. This becomes an antecedent basis which can perform encryption communication exceeding the border lawfully. When transmitting, it is not necessary to check whether the addressee has a key to a decryption. It is because it will end if the addressee receives the data are enciphered and it decrypts later then.

Vidrascu col.6 line21-31
When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.
The transmission is next sent to the interface 31;
When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30.

col.6 line21-31

An enciphered program is in interface 30, a decryption program is in interface 31, it is a talk in the case of decrypting what was enciphered with interface 30 with received interface 31, and the existing place is different.

The response is not considered as the place of interface 30 is equipped with an enciphered program and a decryption program. Since it stands to reason that it is decrypted after the enciphered data is received, the necessity that the address, encryption software, an enciphering key, decryption software, and a decryption key must be added to the item of a table from this idea does not come out.

And the way of thinking which uses an algorithm which is different by encryption and decryption since only DES is considered as a cipher system, or enciphers at the time of transmission, and does not encipher reception is not included at all.
Since DES is a symmetrical key method, the key is common. Two keys are investigated also in the time of transmission. Two keys are investigated also in the time of reception. This means that the reply to A from B is similarly enciphered by DES because two keys exist, when the transmission to B from A is enciphered by a DES method. And even if it investigates that the same key exists in two hands, it is meaningless because it is enciphered using the same key.
Required one,
It is checking the encryption software which is needed when A's transmits to B, and an enciphering key. What is necessary is to encipher and transmit, and just to send with a plaintext, if it does not exist if it exists.
When receiving the data from B, it is investigating the existence of decryption software and a

decryption key corresponding to B. What is necessary is just to consider that it is sent with a plaintext, when it does not exist.
Because only a plaintext may be able to send B to A even if A can send a cipher to B. Since there is no consideration to such a situation, it is not going to investigate whether both encryption software and decryption software are in interface 30.
Of course, it must be thought that these two are the different encryption software and decryption software of an algorithm.

There is also another problem.

It came to be able to use the mail of "user@domain" form in 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "user12@domain1" have the same mail server which having a same IP address.
Moreover, in the communication from "user11@domain1" to "user21@domain2" and also from "user12@domain1" to "user22@domain2", the same encryption will be done.

It came to be able to use the mail of"User@domein" form in 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "HYPERLINK mailto:user@domein" have the same mail server which having a same IP address.
Moreover, in the communication from "user11@domain1" to "HYPERLINK mailto:user@domein" and also from "HYPERLINK mailto:user@domein" to "HYPERLINK mailto:user@domein", the encription will be done.

There are very many people using the same provider, and the users using the same IP address increase in number inevitably. Therefore, with the same method and the same enciphering key, a lot of data will be enciphered and it will transmit. This becomes a big cause by which a code will be decoded.
The response between IP addresses means the response between mail servers. Now, it will be only a manager of a mail server that encryption software can be changed. There are much people using the same provider's mail server. The group of the people using two mail servers of them will use the same encryption software. Now, each user of an E-mail cannot change encryption software freely. It must be able to change, immediately after decode art is found.
In order for the user of an E-mail to choose the encryption software which he can trust and to use freely, encryption software and an enciphering key must be able to be set up not according to the response of an IP address and an IP address but according to the response between mail addresses. If he purchases new machinery, it should enable it to change into a firmer code. If this is not made, cryptocommunication with those who live in the foreign country where legal limits differ will be the origin of a trouble. For realizing people of various countries registered into their own address book, and smooth cryptocommunication, the user of an E-mail has to be able to set up finely freely in his own address book.
It is not written that choosing the key of a code to compensate for an IP address sets and chooses encryption software as an IP address although written. The encryption software which appears in this is only DES, and description which uses the code of an algorithm which it is at the transmission [to the same partner] and reception time from the same partner, and is different does not exist.
It is not written that decryption software and a decryption key are set and chosen as a partner's IP address.

This method includes the further trouble. It is the point of dividing and considering a safe network and the network which is not safe, and trying to perform encryption and a decryption on the boundary. The problem is whether the local network made safe exists. The customer data of a company are sold and bought at a high price. A possibility of intercepting the packet which is flowing without enciphering personnel in the local network made safe, and selling outside is also fully considered. Most leaks of information are the forms where human being in the company

makes confidential information flow into external. Therefore, the boundary of a safe network and the network which is not safe etc. does not exist. It needs to be enciphered when coming out of its own personal computer. Therefore, management of encryption software must be directly carried out with each personal computer, and everybody need to have the address book extended into its own personal computer.

col.12 line 1-11
if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.
verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and
if the first and second keys exist, deciphering a received message with the second key and otherwise reject a received message.

Checking here is not checking existence of encryption software only in existence of a key. It is an item which must be checked if it may not encipher as the case where use many kinds of encryption software, or it enciphers. It is because it does not consider saying that encryption software considers only DES and the thing without the description will change the kind of encryption software in the time of transmission and reception (for example, DES) (for example, AES) at all.
If change of encryption software is considered, the description about whether what we do with change and registration of encryption software by who determining in what kind of position will be required.
It is a problem that they are ″keys″ and the plural form. If transmission and reception are separated, the number of keys required for encryption of transmission will be one. Only existence of an enciphering key and encryption software should be considered at the time of transmission.
Since the person of a position who cannot transmit the enciphered data is also needed even if the enciphered data is receivable.

By the applicant's method, it is an enciphering key and the encryption software which are checked at the time of transmission. Because those who return only the data which is not enciphered are also needed even if it receives code data.
Naturally at the time of reception, only existence of a decryption key and decryption software is considered. It is because the data enciphered to the person may be unable to be transmitted even if the enciphered data is receivable from the person. The existence of a country by which offender treatment will be carried out only by receiving the enciphered data is also considered. Although code mail is unreceivable, it thinks, also when code mail can be transmitted.
Therefore, if only an enciphering key and encryption software are not investigated at the time of transmission and it is not made not to investigate only a decryption key and decryption software at the time of reception, the cryptocommunication beyond the border becomes impossible.
The method that registration and change of a user are freely attained in a cipher program is not shown at all by this literature. By batch management, privacy cannot be protected with a mail server.

col. 6 lines 23-27
if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.

Fig. 13 95, 102
1st ans 2nd keys exist?

Fig. 13 97
Encipherment of message to be sent by 1st key

Fig. 13 104
Decipherment of received message with 2nd key

Vidrasuc has further problems as bellow.
How is it checked where this key 2 exists?
(1) Is a certain thing checked [both of] to a sending person?
(2) Does it check that there is key 1 to a sending person and there is key 2 to an addressee by another communication?
When transmitting data toward many IP addresses, it will stop having a realistic meaning by the method of (2).
In the case of (1), a problem becomes large more. In a common key system, an enciphering key and decryption key is also the same. It does not become the guarantee of the addressee having the key now that there were the two same things. In a public key system, the addressee must not hand others the secret key which he has. It does not become that the sending person has not only in the public key which the addressee created. A method which has a secret key which a sending person uses by decryption will make cryptocommunication in a public key system spoilt. This is the absurd method.

The method which the applicant propose also from this is the outstanding method of being equivalent to the public key system which is important in a present-day cipher system unlike how to propose Vidrascu.
The place which is not studying in what kind of form encryption software and a key should be registered has a fault of Vidrascu.

After it is decrypted, it is clear that its it must be again changed into voice, even if who considers the enciphered sound signal.
However, I think that what it was considered that would make the cipher system at the time of appeal and the cipher system when replying to it into another thing is big progress.
Although this cannot transmit the enciphered voice, the foundation of the lawful cryptocommunication of those who live in the country for which reception of the enciphered voice is possible, and those, who live in the country which can use the telephone enciphered freely, is given.
In communication of those who live in the encryption algorithm which can be used at the time of transmission, and the country which has a limit in the length of a key, and those who live in a country without a limit, it is at the reception and transmitting time and cryptographic algorithm and a key must completely be able to be set up independently.

As for Fig.3 and Fig.6 of Scheir

In this drawing, only the response of encryption software is written to be a key (place which can be interpreted as the telephone number). The portion of an enciphering key is required and the item of decryption software and a decryption key is indispensable further. Without these items, not every user of an E-mail can use encryption software freely, and he can respond to neither the difference in the law for every country, nor the difference in the rule for every organization finely.

Scheir col. 10 line 1
“Receiving device 78 includes a copy of table 80”

This means that the whole has a common encryption software database, and shows that it is a method which cannot cope with a copyright problem, a law problem, each one of economic matters, etc.
That a communications partner has a copy, if a tapping person can participate in this cryptocommunication network, a tapping person will also have a copy. This will deny the effectiveness of a cryptocommunication network itself.
The extended address book must never be copied. It is important for the safety of the whole

communication network to minimize the versatility of the cryptographic algorithm as the whole cryptocommunication network and the cryptographic algorithm accumulated by the individual. By the method of Scheir, although it is impossible, if it is my method, this will become possible.

Schier , col.9 line67　col.10 line1-7
The receiving device 78 includes a copy of table 80. The copy of table 80 is then used at step 132 to retrieve the first indicated encryption algorithm from the encryption algorithm column 84. This encryption algorithm is then loaded into the encryption decryption engine 30 and is executed by central processing unit 20 to encrypt outgoing communication and decrypt incoming communication at step 134.

This trouble is having a copy of table 80. It is that the decryption software specified by the same number as the encryption software specified by the number of the encryption algorithm indicated on the table exists in both 78 and 72.
Mr. B and Mr. C will also have the same table, this will become ... etc., if Mr. A and Mr. B have the same table, and all the members will have the same table.
It contains a problem in some respects that those [all] that participate in a cryptocommunication network have a common table.

1. Can this Table Perform Addition, Deletion, and Updating?
2. Who Makes Decision of Addition, Deletion, and Updating?
3. How is Added Software Distributed?
4. Who Pays Royalty how?
5. Doesn't Distribution beyond Country Become Criminal Act?
6. When a table changes to a powerful new code, can the owner of old machinery use?
7. Why Do People with New Machinery Have the Need of Saving Old Code Which Cannot be Used?

If a table is shared, such a problem will be able to be thrust before. Sharing of a table must be given up. On the contrary, never sharing is a basic principle which builds a firm cryptocommunication network.
It solves the above-mentioned conflict that only its part manages the address book in which everybody were extended.

Using a copy will share the same encryption software as the table with all the same machinery that participates in this cryptocommunication network, and the same decryption software.
Since this encryption communication network accepts intervention of outdated machinery, the weak encryption which is used also by low count ability will be used.
Since all the machinery has a copy, if the number of the machinery which participates in the encryption communication network increases, changing into new cryptographic algorithm will become very difficult. Because, it is because there are those who can pay the royalty of encryption software or decryption software, and those who cannot pay, and the law of a different country may allow the activity of the algorithm, or may not be allowed. There are some countries which have forbidden carrying out powerful encryption software abroad, and there are also some countries which have allowed carrying out freely. If a copy is distributed on the Internet, it will become a criminal act in many cases.
Although a problem is not used in a certain country about the data encryption to receive, it thinks, also when applying a limit to the data encryption which transmits.
If it does so, in the country in which Mr. A lives, a code can be used freely, and although reception is free, when transmitting, only easy encryption may be allowed in the country in which Mr. B lives. At this time, the encryption software of Mr. A's address book registers the newest powerful thing, and can register what also has a long enciphering key. However, the decryption software of Mr. B's place must be easy, and the length of a decryption key must be short.
If the table and encryption software of encryption of Mr. A are sent to Mr. B, he will be an offender only by Mr. B using it. It is actual that legal limits differ about a code. In order to build as

effective a cryptocommunication network as possible in this, the extended address book (telephone number book) which everybody manage uniquely is required.
There is also a country which is asking for submitting a decryption key to the public institution of a country. If a copy is used, the country which submitted the key can decode all the communications. Since the information on such an accumulated decryption key and an enciphering key can be sold at a high price, worries about the leak of information from the interior of a public institution also take place. If those [all] that participate in a network share encryption and a decryption, it will have the capacity which all the members decode.
For the above reason, this method is not fit for internationalized communication. A copy must be given up. However, this difficulty is solvable if the idea of an address book that I was extended is used.
[if encryption software is only called from the telephone number, are an easy talk, but]
Encryption software is divided into an encryption portion and a decryption portion, and it is sure that it is the most important point to extend an address book in the form suitable for it, and to have enabled it to register encryption software, an enciphering key, decryption software, and a decryption key. This point is not described at all. Never don't perform description which distributes a copy so that it may not destroy a cryptocommunication network itself.

By the applicant's method
Mr. A and Mr. B buy new telephone, and suppose that Mr. C is continuing using old telephone. At this time, by the method which I propose, the encryption software and enciphering key which are registered into the line of Mr. B of the address book which Mr. A has can register the newest thing, and can use it.
the measure not using a cipher system easy [since Mr. C has only old telephone] also when performing cryptocommunication with Mr. A, and also when performing cryptocommunication with Mr. B. Since a decryption and encryption take time too much by old telephone, a firm new code cannot be used. Therefore, old, easy encryption software must be registered into the line of Mr. B of the address book in which Mr. C was extended.
Therefore, at all, the address book in which Mr. A was extended, and the address book in which Mr. C was extended are another, and are not a copy.
The way of intensity as the whole cryptocommunication network where the individual can be made to do a setup of encryption and a decryption according to the performance of the telephone which he has, and the performance of the telephone which its communications partner has, respectively improves.
Even if it sees from this point, five items, the address, encryption software, an enciphering key, decryption software, and a decryption key, are needed for the item of an address book or a telephone number book.

Each one of address books (signal book) are managed to everybody. Communalization is not carried out. This is important. If it carries out like this, new machinery can make the environment where the force can be demonstrated immediately. Old machinery must not threaten the safety of a cryptocommunication network. It is important that the contents of the address book extended in the form which suited everybody's situation of its communications partner can be adjusted. [oneself and]
Even if it sees from this point, the extended address book improves the safety of cryptocommunication by leaps and bounds, and it is the method of moreover also solving the legal trouble in international cryptocommunication, and is sure that prior art is excelled.
It is because it thought that that this became possible would encipher by respectively separate encryption algorithm by dividing into two, the communication of information which had the directivity from Mr. A to Mr. B for the conversation of Mr. A and Mr. B, and communication of information with the directivity from Mr. B to Mr. A.
I hear that the conversation {A, B} of A and B was decomposed into two, (A->B) and (B->A), and separate encryption was introduced to each when symbolizing.

[Regarding item 19]

Schier , Fig.3 and col. 4 lines 30-56

FIG. 3 illustrates a selected portion of one embodiment of encryption table 28 which was disclosed with reference to FIG. 2 previously. Encryption selection table 28 comprises a key column 34, a first algorithm column 36, a second algorithm column 38 and a third algorithm column 40. In operation, a particular device as device 14 would have the capability of performing a number of distinct encryption processes. For example, device 14 may be able to perform five different encryption techniques. An encryption key in key column 34 is then used to access a particular row which specifies a particular encryption technique in each of columns 36, 38 and 40. A message to be sent by device 14 or received by device 14 can then be encrypted or decrypted using the technique specified in the row in the order specified in the row. For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4. In this manner, a number of relatively simple encryption steps can be sequentially applied to a message to greatly enhance the security of the message. A person attempting to intercept and wrongfully decrypt the message would have to discern not only the various kinds encryption used, but also the order in which the technique were used.

Further problems bellow

For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1.

Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.

The encryption algorithm in device 14 and the decryption algorithm in device 14 are not divided. This shows that the symmetric property of cryptocommunication is accepted unconditionally. Existence of those who can return only a plaintext even if code data is receivable, and those who can return only the data which carried out weak encryption even if the data which gave the powerful code is receivable is disregarded.

Everyone is not the same position. The structure which can adopt a different algorithm must be prepared in encryption and a decryption.

A legal problem does not allow having common encryption software in many cases. Cryptocommunication must not make an offender.

It is required for a correspondence table to provide a decryption portion in the form where it became independent. The method of Schier cannot solve this problem.

[Regarding item 20]

Even if it Interprets it as Individual Having IP Address even if, as for the Method of Vidrascu, Table Structure Which Registers Checking Two Keys of Fig.13, Key, and Encryption Software is not Specified,

It cannot respond to a present-day public key cryptosystem at all, and it must be told to it that it is an outdated method.

It is clear not to give the cryptocommunication network in which such a thing is safe and lawful, and my method of building a certainly firm and lawful cryptocommunication network is completely heterogeneous. Therefore, the contents which Vidrascu showed do not seem to become a reason for refusal of the present invention of me.

The origin which produces such a difference is in the place which says whether the individual has the extended address book and whether the item of a name, the address, encryption software, an enciphering key, decryption software, and a decryption key exists as an item in it further.

This point is a nucleus portion of the method which the applicant propose. Vidrascu is not solving at all about this point.

[Regarding item 21]

If the method of Vidrascu, Leppek, Collins, and Schier is unified, it is not so difficult to think of determining the key to encryption from a mail address, or determining a series of encryption software used by multiplex encryption from the telephone number. This is as an examiner being said.
They explain the process in which what A sends to B is enciphered, when considering communication of A and B. And a situation in case received B decrypts is explained.
The conversion table which they use is two, the table which A uses by encryption, and the table which B uses by decryption.
If it does not consider, it does not sometimes change [two]. It is a table used in order that B may encipher what sent to A, and a table used in order that A may decrypt what received from B.
The way of thinking referred to as dividing intentionally what used when enciphering what puts in the portions of decryption software and a decryption key and A sends into a conversion table at B, if this is not considered, and the thing used when A decrypts what sent from B, and putting it in is not produced.
Since this is registered independently, it is not meaningful if the program of a different algorithm cannot be used. Therefore, the cryptocommunication in this case becomes asymmetrical. It comes out as a theoretical possibility to say that the algorithm with which all the enciphered programs used with a cryptocommunication network differ may be used. This teaches that there is no meaning using the database of encryption software.
This is never a matter thought of ordinarily. It is a matter which is not thought of unless it throws away symmetric property and a code database. Therefore, even if it gathers up the method of Vidrascu, Leppek, Collins, and Schier, the structure of the extended address book does not come out.
If there is no strong will that using a code database by existing cryptocommunication, or being premised on the property of symmetrical cryptocommunication notices many problems included, and it solves these, the way of thinking of the extended address book does not have a birth lever.
Although the item was only increased, the trouble that just this structure is inherent in a cryptocommunication network is solved at once. Therefore, it is sure that there is meaning of enough as an invention.
It sees in detail below.

Vidrascu, col.6 lines 21-27
When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.
The transmission is next sent to the interface 31;

The meaning of an IP address becomes the beginning with a problem. This is whether to be able to specify the individual who participates in a cryptocommunication network.
The document of Vidrascu was submitted in 1993. Before

However, the mail of “user@domain” form has been used since 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "user12@domain1" have the same mail server which having a same IP address.
Moreover, in the communication from "user11@domain1" to "user21@domain2" and also from "user12@domain1" to "user22@domain2", the same encryption will be done.

However, the mail of “user@domain” form has been used since 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "HYPERLINK mailto:user@domein" have the same mail server which having a same IP address.

Moreover, in the communication from "user11@domain1" to "HYPERLINK mailto:user@domein" and also from "HYPERLINK mailto:user@domein" to "HYPERLINK mailto:user@domein", the encription will be done.

There are very many people using the same provider's mail server, and the users using the same IP address increase in number inevitably. Therefore, with the same method and the same enciphering key, a lot of data will be enciphered and it will transmit. This becomes a big cause by which a code will be decoded.
The response between IP addresses means the response between mail servers. Now, it will be only a manager of a mail server that encryption software can be changed. There are much people using the same provider's mail server. The group of the people using two mail servers of them will use the same encryption software. Now, each user of an E-mail cannot change encryption software freely. It must be able to change, immediately after decode art is found.
In order for the user of an E-mail to choose the encryption software which suited itself and to use freely, not according to the response of an IP address and an IP address but according to the response between mail addresses, encryption software, an enciphering key, decryption software, and a decryption key must be able to be set up freely.
If this is not made, cryptocommunication with those who live in the foreign country where legal limits differ will be the origin of a trouble. For realizing people of various countries registered into their own address book, and smooth cryptocommunication, the user of an E-mail has to be able to set up finely freely in his own address book.
The portion of "if the keys related to the IP addresses of the senderand of the receiver are found" poses a problem. Considering a public key system, the public key for encryption and the secret key for a decryption will be left [both] to the hand of the mail server manager of the sending person side. Of course, since a secret key is an important thing which receives encryption data and decrypts it, it must be placed to an individual. Of course, it cannot put on the mail server used as a receiving side, either. Don't put on any hands other than the final addressee of the enciphered information. The meaning of public key encryption is lost.
It is the same even if it thinks that an IP address expresses an individual. Considering public key encryption, a system which should check only the public key used by encryption and checks even a secret key is completely outdated.
In order to be able to realize this certainly, the item of a name, the address, encryption software, an enciphering key, decryption software, and a decryption key must be made to the extended address book.
And although encryption software and an enciphering key are registered, decryption software and the decryption key must make the system which runs also by a situation [a situation] which is not registered exactly.
Of course, although a reverse situation, i.e., encryption software and an enciphering key, is not registered, decryption software and the decryption key must make the system which runs also by a situation which is registered exactly.
The system which runs also by the situation where the algorithm of encryption software completely differs from the algorithm of decryption software, exactly must be made.
In order to realize this, the item of the extended above-mentioned address book is needed. Of course, it does not become a practical system although the item of a name is logically unnecessary.

Vidrascu, col.6 lines 27-36
When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30;

Here, if the meaning of an IP address is an IP address of an ordinary mail server, there is a

decryption key to a server manager and a server manager will know all individual secrets.
Even if it assumes that all the individuals have an IP address, in order to decrypt the enciphered data, it is enough if there is a key to a decryption.
It is not written that choosing the key of a code to compensate for an IP address sets and chooses encryption software as an IP address although written. The encryption software which appears in this is only DES, and description which chooses other encryption algorithms does not exist.
This method includes the further trouble. It is the point of dividing and considering a safe network and the network which is not safe, and trying to perform encryption and a decryption on the boundary. The problem is whether the local network made safe exists. The customer data of a company are sold and bought at a high price. A possibility of intercepting the packet which is flowing without enciphering personnel in the local network made safe, and selling outside is also fully considered. Most leaks of information are the forms where human being in the company makes confidential information flow into external. Therefore, the boundary of a safe network and the network which is not safe etc. does not exist. It needs to be enciphered when coming out of its own personal computer. Therefore, management of encryption software must be directly carried out with each personal computer, and everybody need to have the address book extended into its own personal computer.

Vidrascu, col.6 lines 21-36

The examiner is checking that only DES appears here. The literature of Leppek in which much encryption software and decryption software appear is quoted.
Although it stands to reason that much encryption software appears in multiplex encryption, or much decryption software corresponding to it is treated,
As DES is used for important one when Mr. A sends to Mr. B, and AES is used for decrypting what Mr. A received from Mr. B,
An address book which can register the encryption software of an algorithm which is different by the encryption item and a decryption item and which was extended must be considered.
This gives the database of encryption software common to the whole cryptocommunication network as unnecessary.
What exports encryption software illegally can be avoided now by this. If he purchases new machinery, it will enable it to change into a firmer code immediately.

Vidrascu, col.12 line 1-11
if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.
verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and
if the first and second keys exist, deciphering a received message with the second key and otherwise reject a received message.

Checking here is not checking existence of encryption software only in existence of a key. It is an item which must be checked if it may not encipher as the case where use many kinds of encryption software, or it enciphers. It is because, as for the thing without the description, encryption software considers only DES and change of encryption software is not considered at all.
If change of encryption software is considered, the description about whether what we do with change and registration of encryption software by who determining in what kind of position will be required.
The 1st key is used for encryption and the 2nd key is used for a decryption. It does not become the guarantee of there being the 2nd key to an addressee that there is the 2nd key to a sending person.

In the case of public key encryption, a sending person cannot have the 2nd key. Therefore, by this method, it cannot respond to public key encryption.
Only existence of an enciphering key and encryption software should be considered at the time of transmission. Considering only existence of a decryption key and decryption software, at the time of reception, it is enough.

Leppek, col.2 lines 19-23
The fundamental machine of the virtual encryption scheme of the involves the generation of a sequence of the access code, with immediately successive ones of the access code of the sequence being different from one another.
Leppek, col.2 lines 51-55,
The order of the encryptors within the sequence to which the data is applied may very as desired, and the sequence may 'toggle' or switch back and forth between the same set of encryption operators as part of its overall encryption flow.
Leppek, col.4 lines 14-17
The encryption routines 110, in and of themselves, need not be any particular type of encryption algorithm and may be conventional encryption operations, such as PGP, DES, etc. routines, as non-limiting examples.

It considers changing the turn of encryption.
He will explain that the train of a different enciphered program is specified, if encr.key differs.
He is not necessarily explaining that a setup of supervisor encryption manager is changed and the train of a different enciphered program can be generated to the same encr.key.
About the updating method of encryption operator database etc., it is not touching at all.
Cryptographic algorithm will become old if no less than three year passes now. It is clear if DES, Triple-DES, AES, and the process in which it has changed are seen. Therefore, it does not become a practical system if renewal of the database to be used is not considered.

It is possible to register the encryption software of a different algorithm into the line of the different address by the method of using the address book in which I was extended. Of course, the order to apply can also be set up freely.
Renewal of the encryption software used by [new] purchasing that it is soft and sending a table against a soft direct rewriting and encryption portion can be performed.
Of course, the turn which applies the encryption software registered into a certain line can also be changed easily. What is necessary is to write a table directly and just to change.
Of course, also about a decryption portion, change is possible and, moreover, it can change independently with an encryption portion.
Even if it sees from this point, it is sure that my method is a method which was excellent although it was simple. If a database is used, it will become spoilt [a special sequence change].

Collins, Col.5 lines 43-63
The Internet e-mail gateway 240 converts the SMS message 100 to an Internet e-mail message having the format 150. To accomplish this, the Internet e-mail gateway 240 determines an Internet e-mail address for the cellular telephone 210 and for the Internet station 280. Regarding the Internet e-mail address for the cellular telephone 210, the Internet e-mail gateway 240 adds the domain name for the Internet e-mail gateway 240 to the SMS address for the cellular telephone located in the sender SMS address field 110 of the SMS message. To determine the Internet e-mail address for the Internet Station 280, the Internet e-mail gateway 240 uses a messaging service lookup table having a first column with a list of SMS addresses, where each SMS address corresponds to an Internet e-mail address in a second column of the lookup table. The Internet e-mail gateway 240 locates the SMS address of the Internet station 280 in the first column of the lookup table. The Internet e-mail address for the Internet station 280 is retrieved from the second column of the lookup table at a location corresponding to the Internet station SMS address of the first column.

Here, the SMS address, conversion of Internet e-mail address, and changing portable mail into an Internet mail by it are only described. The mechanism in which the portable mail accepted only in the narrow range circulates in the whole world through the Internet by this is solved.

Further,
Schier, Fig.3
Although it was alike, and it was related and the judgment referred to as that an examiner uses a table for speedup of a data access is expressed, the address book extended even if it changed the train on the left of this drawing into the train of the mail address is not obtained. Now, the structure of the extended address book cannot be discovered. Because, the method of enciphering signal transfer in case Mr. A calls to Mr. B in communication of Mr. A and Mr. B, It is because it is not solved why it must be made separate, although it is also possible to make the same the method of enciphering signal transfer in case Mr. B replies to Mr. A corresponding to it and it is easier.
This is the cause, and advanced technology includes various troubles and has made development of cryptocommunication difficult. Though development of cryptocommunication is desired, only a thing of structure which blocks it is offered. Assuming the symmetric property of cryptocommunication and the way of thinking of using a code database are the origins in question.
It is difficult to notice the need for throwing away the symmetric property and database. This is just a nucleus portion of the present invention.
For building a cryptocommunication network firm as a whole, corresponding to improvement in the count ability of the difference between the problem of copyright, and the law for every country, the difference in the count ability of the machinery which an individual owns, the decryption art developed newly, and the computer used for decryption etc.,
Just the method which considers only the thing of the mechanical performance in which he has [the individual] the extended address book provided with the item encryption software, an enciphering key, decryption software, and a decryption key's, and each individual's communications partner, and can be developed freely is required, and various problems can be solved by this.
It became four items of need for an address book address book just because it thought that two different cipher systems were used for communication of Mr. A and Mr. B, and the solving means of the copyright problem by separation of the encryption portion of software and a decryption portion or an illegal copy problem has been offered by it. It is sure that this point is big progress.
Such a point of argument is not solved at all in the literature in which the examiner was shown. Therefore, I think that my method has suitable unobviousness in being patented.

[Regarding item 22]

It is the art from ancient times which it treats as an attached file, and suitable conversion of base64 etc. is carried out, and is transmitted about this in order to send binary data by E-mail so that an examiner may be said.
It is because he wanted to show that it must be made to have to agree in the existing abbreviation of a communication network since the data which having described this enciphered differs in the data and the property of a basis. Since it was not not much clear, I wanted to emphasize the need by telephone.
Since what cannot transmit is an obvious thing if data does not suit the protocol of a communication network, it may be an item which should be deleted.

23. The literature of Leppek does not explain how object 160 in Fig.2 is generated from encr.key.
The way the structure of supervisory encryption manager is indefinite and generates object 160

from a key (encr.key) is not described at all. encr.key is what -- it is not explaining a thing.
When a new participant participates in this cryptocommunication network, can supervisory encryption manager be responded?
Supposing it can respond, what kind of train 160 will be generated to the participant who added newly?
How does a new participant obtain the encryption software registered into encryption operation data base?
How does a new participant obtain the encryption software registered into decryption operation data base?
By this method, problems about copyright, such as a problem and an import restriction of encryption software, will remain as an unconquerable subject.
In a common database, the difference in the law for every country about a code is unconquerable. Therefore, common encryption operation data base The way of thinking referred to as using is a method which does not suit international communication.
Table 100 includes the big trouble by itself. As for this database, separation of encryption software and an enciphering key is not made. Therefore, when it thinks as a thing having contained the enciphering key, the number of the encryption software which must be registered into a database increases extremely.
For example, in RSA, although the length of the key made safe is a grade in 2048 bits, a length of about 4000 bits is needed immediately. When it thinks simply, the number of a 4000-bit key is a 1200th power individual of the 4000th power of 2, i.e., about 10. The present hard disk is about 100 G bytes. This is an 11th power byte of 10. A safe database cannot be built however he may do his best.
The database as the whole must be given up and an individual has to consider it focusing on what can manage freely only the portion which he uses. The extended address book which the applicant propose solves such a problem.

Leppek, col. 4 lines 24-52
As described briefly above, the fundamental control factor used by the virtual encryption scheme of the invention is the fact it produces a sequence 160 of access address codes 120-i, such that immediately successive code 120-i and 120-j in the assembled code sequence are different from one another. Thus, for an arbitrary plurality N of respectively different data enciphering routine or operator entries 110-1, 100-2, 100-3,…,100-N, there will be N associated access address codes 120-1, 120-2, 120-3,…,120-N.
In accordance with the invention the supervisory encryption assembly manager 130 is supplied with an encryption driver or key 170 comprised of a sequence of M access code entries made up of K (at least two and up to all N) address code entries 120 for the encryption operators 110 stored in the database 100. M may be any number equal to or greater than two. Thus, at a minimum, address code sequence 140 would be defined by only two respectively different ones 120-i and 120-j of the N available codes, so that M would be equal to two, regardless of N. Even if N is only two, M is still unbounded, since it may comprise an alternating sequence of arbitrary length. Namely, where N=2, the database 100 would have only two entities 120-1 and 120-2. In this case, an encryption control access code sequence of length M could be generated as the alternating sequence …, 120-1, 120-2, 120-1, 120-2, 120-1, 120-2,…, up to M entries, where M>2. What is important is that the respective codes of any successive pair of codes differ from one another.

Here, the situation of communication by multiplex encryption is only described. It is indifferent to the trouble of an encryption database.
The constitution method of a code database and the method of the current update are not described at all. If no less than three year passes, it will be the encoding technology which becomes old. This database becomes old, and those who participate in this cryptocommunication network newly will demand renewal of a code database, when it stops becoming useful. It was already said that it is difficult to reply to this.
And a new participant will refuse to put the never used old software on his own personal computer. Even if it makes the train of 160 with much trouble from this, since the contents of the database are different, in it, it cannot even perform decrypting what was received.

It becomes absurd difficult work from a legal problem to distribute the updated database to the whole cryptocommunication network. Just as it distributes on the Internet, it is considered that arms were exported and it becomes an offender. When a duplicate is distributed in large quantities, copyright and a patent right will be invaded.
Of course, if it is the encryption software of a toy which does not have a meaning used being too weak, it may be able to distribute, but even if it distributes this, nobody uses.
This method stands on the premise made the mistake in referring to as being able to distribute the code database which consists of a powerful code to the whole communication network. Here which uses making a code database and choosing here from them is a big problem. Everybody should have only a part which needs each. What is necessary is to have only a possible thing legally. Don't make a code database. Progress of encoding technology is quick. The structure which can be updated rapidly must be made from judgment of it. If it is the extended address book which the applicant propose, these problems are solvable, and it can build without touching a law with a cryptocommunication network powerful as a whole.

[Regarding item 24]

To be sure, as for the sequence of encryption, and the sequence of a decryption, it is obvious that you must be reverse so that an examiner may be said.

For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.

1. A encrypting the data to be sent to B.
2. B decrypting the data from A.
3. B encrypting the data to be sent to A.
4. A decrypting the data from B.

Schier considers 1 and 3 the same way. This will assume the symmetric property of cryptocommunication.
When [however,] Mr. A transmits the data by which multiplex encryption was carried out to Mr. B Some cryptographic algorithms to be used and the cryptographic algorithm used when Mr. A decrypts the data sent by Mr. B need to be completely able to choose another thing, and the structure of an address book where it was extended for realizing it is by no means obvious. The place which can change the sequence independently applied about the portion of encryption and the portion of a decryption in such an extended address book (in the case of multiplex encryption), respectively serves as a new portion.

[Regarding item 25]

Claim again. Only the method which the applicant propose instead of the method of Vidrascu solves various troubles which a cryptocommunication network holds, and can build a firm and brief cryptocommunication network.
The extended address book is absolutely required.

[Regarding item 26]

Although indication here overlaps with indication by 21, carry out it to stating again.

Vidrascu, col.6 lines 21-27

When IP data transmission transporting a TCP or UDP protocol is received (normally plane) on the interface 30 (See FIG1), and if the keys related to the IP addresses of the sender and of the receiver are found, a part of the user data of this transmission is enciphered with a DES algorithm by using a key related to the IP address of the sender.
The transmission is next sent to the interface 31;

The meaning of an IP address becomes the beginning with a problem. This is whether to be able to specify the individual who participates in a cryptocommunication network.
The document of Vidrascu was submitted in 1993. Before

However, the mail of "user@domain" form has been used since 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "user12@domain1" have the same mail server which having a same IP address.
Moreover, in the communication from "user11@domain1" to "user21@domain2" and also from "user12@domain1" to "user22@domain2", the same encryption will be done.

However, the mail of "user@domain" form has been used since 1986. IP address corresponds to "domain" after the mark "@". Therefore, both "user11@domain1" and "HYPERLINK mailto:user@domein" have the same mail server which having a same IP address.
Moreover, in the communication from "user11@domain1" to "HYPERLINK mailto:user@domein" and also from "HYPERLINK mailto:user@domein" to "HYPERLINK mailto:user@domein", the encription will be done.

There are very many people using the same provider's mail server, and the users using the same IP address increase in number inevitably. Therefore, with the same method and the same enciphering key, a lot of data will be enciphered and it will transmit. It becomes a big cause by which a code will be decoded.
The response between IP addresses means the response between mail servers. Now, it will be only a manager of a mail server that encryption software can be changed. There are much people using the same provider's mail server. The group of the people using two mail servers of them will use the same encryption software. Now, each user of an E-mail cannot change encryption software freely. It must be able to change, immediately after decode art is found.
In order for the user of an E-mail to choose the encryption software which suited itself and to use freely, not according to the response of an IP address and an IP address but according to the response between mail addresses, encryption software, an enciphering key, decryption software, and a decryption key must be able to be set up freely.
If this is not made, cryptocommunication with those who live in the foreign country where legal limits differ will be the origin of a trouble. For realizing people of various countries registered into their own address book, and smooth cryptocommunication, the user of an E-mail has to be able to set up finely freely in his own address book.
It becomes a problem,
It is a portion of "if the keys related to the IP addresses of the sender and of the receiver are found." Considering a public key system, the public key for encryption and the secret key for a decryption will be left [both] to the hand of the mail server manager of the sending person side. Of course, since a secret key is an important thing which receives encryption data and decrypts it, it must be placed to an individual. Of course, it cannot put on the mail server used as a receiving side, either. Don't put on any hands other than the final addressee of the enciphered information. The meaning of public key encryption is lost.
It is the same even if it thinks that an IP address expresses an individual. Considering public key encryption, a system which should check only the public key used by encryption and checks even a secret key is completely outdated.
In order to be able to realize this certainly, the item of a name, the address, encryption software, an enciphering key, decryption software, and a decryption key must be made to the extended address book.

And although encryption software and an enciphering key are registered, decryption software and the decryption key must make the system which runs also by a situation [a situation] which is not registered exactly.
Of course, although a reverse situation, i.e., encryption software and an enciphering key, is not registered, decryption software and the decryption key must make the system which runs also by a situation which is registered exactly.
The system which runs also by the situation where the algorithm of encryption software completely differs from the algorithm of decryption software, exactly must be made.
In order to realize this, the item of the extended above-mentioned address book is needed. Of course, it does not become a practical system although the item of a name is logically unnecessary.

Vidrascu, col.6 lines 27-36
When an IP data transmission transporting a TCP or UDP protocol is received (normally enciphered) on the interface 31, and if the keys related to the IP address of the sender and of the receiver are found, a part of the user data of this datagram is decrypted with DES algorithm by using a key related to the IP address of the sender (the same key as that having served in the enciphering). The datagram is next sent to the interface 30;

Here, if the meaning of an IP address is an IP address of an ordinary mail server, there is a decryption key to a server manager and a server manager will know all individual secrets.
Even if it assumes that all the individuals have an IP address, in order to composite-ize the enciphered data, it is enough if there is a key to a decryption.
It is not written that choosing the key of a code to compensate for an IP address sets and chooses encryption software as an IP address although written. The encryption software which appears in this is only DES, and description which chooses other encryption algorithms does not exist.
This method includes the further trouble. It is the point of dividing and considering a safe network and the network which is not safe, and trying to perform encryption and a decryption on the boundary. The problem is whether the local network made safe exists. The customer data of a company are sold and bought at a high price. A possibility of intercepting the packet which is flowing without enciphering personnel in the local network made safe, and selling outside is also fully considered. Most leaks of information are the forms where human being in the company makes confidential information flow into external. Therefore, the boundary of a safe network and the network which is not safe etc. does not exist. It needs to be enciphered when coming out of its own personal computer. Therefore, management of encryption software must be directly carried out with each personal computer, and everybody need to have the address book extended into its own personal computer.

Vidrascu, col. 6 line 21-26
The examiner is checking that only DES appears here. The literature of Leppek in which much encryption software and decryption software appear is quoted.
Although it stands to reason that much encryption software appears in multiplex encryption, or much decryption software corresponding to it is treated,
As DES is used for important one when Mr. A sends to Mr. B, and AES is used for decrypting what Mr. A received from Mr. B,
An address book which can register the encryption software of an algorithm which is different by the encryption item and a decryption item and which was extended must be considered.
This gives the database of encryption software common to the whole cryptocommunication network as unnecessary.
What exports encryption software illegally can be avoided now by this. If he purchases new machinery, it will enable it to change into a firmer code immediately.

Vidrascu, col.12 line 1-11

if the first and second keys exist, enciphering a message to be sent with the first key to obtain an enciphered message, and transmitting the enciphered message by the sending equipment of the first item of equipment and otherwise rejecting the message to be sent.
verifying a presence of a second memory card to authenticate a second operator of the second item of equipment; and
if the first and second keys exist, deciphering a received message with the second key and otherwise reject a received message.

Checking here is not checking existence of encryption software only in existence of a key. It is an item which must be checked if it may not encipher as the case where use many kinds of encryption software, or it enciphers. It is because, as for the thing without the description, encryption software considers only DES and change of encryption software is not considered at all.
If change of encryption software is considered, the description about whether what we do with change and registration of encryption software by who determining in what kind of position will be required.
The 1st key is used for encryption and the 2nd key is used for a decryption. It does not become the guarantee of there being the 2nd key to an addressee that there is the 2nd key to a sending person.
In the case of public key encryption, a sending person cannot have the 2nd key. Therefore, by this method, it cannot respond to public key encryption.
Only existence of an enciphering key and encryption software should be considered at the time of transmission. Considering only existence of a decryption key and decryption software, at the time of reception, it is enough.

Leppek, col.2 lines 19-23
The fundamental machine of the virtual encryption scheme of the involves the generation of a sequence of the access code, with immediately successive ones of the access code of the sequence being different from one another.
Leppek, col.2 lines 51-55,
The order of the encryptors within the sequence to which the data is applied may very as desired, and the sequence may 'toggle' or switch back and forth between the same set of encryption operators as part of its overall encryption flow.
Leppek, col.4 lines 14-17
The encryption routines 110, in and of themselves, need not be any particular type of encryption algorithm and may be conventional encryption operations, such as PGP, DES, etc. routines, as non-limiting examples.

It considers changing the turn of encryption.
He will explain that the train of a different enciphered program is specified if encr.key differs.
He is not necessarily explaining that a setup of supervisor encryption manager is changed and the train of a different enciphered program can be generated to the same encr.key.
About the updating method of encryption operator database etc., it is not touching at all.
Cryptographic algorithm will become old if no less than three year passes now. It is clear if DES, Triple-DES, AES, and the process in which it has changed are seen. Therefore, it does not become a practical system if renewal of the database to be used is not considered.

It is possible to register the encryption software of a different algorithm into the line of the different address by the method of using the address book in which I was extended. Of course, the order to apply can also be set up freely.
New software is purchased and renewal of the encryption software used by sending a table against a soft direct rewriting and encryption portion can be performed.
Of course, the turn which applies the encryption software registered into a certain line can also be changed easily. What is necessary is to write a table directly and just to change.
Of course, also about a decryption portion, change is possible and, moreover, it can change

independently with an encryption portion.
Even if it sees from this point, it is sure that my method is a method which was excellent although it was simple. If a database is used, it will become spoilt [a special sequence change].

Collins, Col.5 lines 43-63
The Internet e-mail gateway 240 converts the SMS message 100 to an Internet e-mail message having the format 150. To accomplish this, the Internet e-mail gateway 240 determines an Internet e-mail address for the cellular telephone 210 and for the Internet station 280. Regarding the Internet e-mail address for the cellular telephone 210, the Internet e-mail gateway 240 adds the domain name for the Internet e-mail gateway 240 to the SMS address for the cellular telephone located in the sender SMS address field 110 of the SMS message. To determine the Internet e-mail address for the Internet Station 280, the Internet e-mail gateway 240 uses a messaging service lookup table having a first column with a list of SMS addresses, where each SMS address corresponds to an Internet e-mail address in a second column of the lookup table. The Internet e-mail gateway 240 locates the SMS address of the Internet station 280 in the first column of the lookup table. The Internet e-mail address for the Internet station 280 is retrieved from the second column of the lookup table at a location corresponding to the Internet station SMS address of the first column.

Here, the SMS address, conversion of Internet e-mail address, and changing portable mail into an Internet mail by it are only described. The mechanism in which the portable mail accepted only in the narrow range circulates in the whole world through the Internet by this is solved.

Keats, col.5 lines 6-8 (5-11)
Each network element has its own SID (maintained for example in a memory element under control of the SID manager) and also maintains a database, table or other suitable structure in another memory element, such as a cache, which maps TIDs to IP address for IP connections from the particular network element to another network element in the network, as described in detail below.

Changing data using table structure occurs from ancient times. An address book, a telephone directory, various mathematical tables, etc. However, various troubles in cryptocommunication are what, and in order to solve them, nothing is described about whether the table of what kind of structure is required.
The table must be not the instrument of speedup but an instrument of a problem solving only. It is decided what kind of item to be needed by the issue which should be solved. In order to realize international cryptocommunication, the item of the address, encryption software, an enciphering key, decryption software, and a decryption key is required.

Here, the trouble which should be solved is checked and the reason which can solve them is explained.
Troubles are enumerated first.
1. The path length into which the signal which is not enciphered flows is not the shortest.
In the network made safe, the data which is not enciphered flows and it is enciphered on a boundary line with the network made not safe. The place which assumes the existence of a network recognized safe is the biggest trouble. It is the feature of a present-day information crime which information reveals from human being inside an organization in many cases.
2. It is enciphered using an algorithm with much same data.
In the structure of an E-mail, many mail addresses correspond to the same IP address. Many people using one mail server are using the same IP address. Therefore, when a cryptocommunication network is built based on the response of an IP address and an IP address, many E-mails will be enciphered by the same method. This serves as a big key of decryption.
3. A tapping person can also encipher the data which he made by the method.

If a tapping person registers himself as a user of the mail server made into the purpose, the data which is useful for decryption will be transmitted, the feature of an algorithm will be caught, and a possibility of succeeding in decode will become high.
4. There is a problem in the determination of the encoding technology to be used.
When new encoding technology is developed, how much do those who are going to encipher the data which manages a mail server and passes through that pay a developer a soft price? There are not so many mail servers. Only a small number of encryption software will be adopted. The developer's development costs will not fully be filled up.
Who has charge in the safety of a code, as for the further problem? There is also a place required as some country and organizations submitting an enciphering key and a decryption key. The meaning of cryptocommunication will be lost if this information leaks.
5. There is a problem in change to the encoding technology to be used.
When using new art, cryptocommunication will become meaningless if the decode method already exists.
Can the effectiveness be proved exactly to a user? The cryptographic algorithm whose existence of a backdoor was being doubted had also existed for a long time. There is also an example in which the method said to be safe with certification has been decoded.
If it decodes before paying high money, updating the encryption software which became old and three year's passing, who will take charge?
6. A fine response is not made to the difference in the law by a country.
Although code mail is receivable for some country and organizations, there can be no regulation which cannot transmit code mail, only the enciphering key and decryption key which were registered can be used, or only weak encryption can be used.
A setup that those who participate in a cryptocommunication network do not do a setup or encryption of encryption finely for every communications partner must be possible. In the response to an IP address from an IP address, this is impossible.

The above trouble is solvable by the method which the applicant propose.
1. Data is enciphered when coming out of a personal computer. The enciphered data is decrypted in a personal computer. Therefore, the communication path into which the data which is not enciphered flows does not exist.
2. It is enciphered using the algorithm with which much data differs.
By the method which I propose, if economical and technical conditions allow, different encryption algorithms to the group of each sending person's mail address and an addressee's mail address can be used.
And the encryption algorithm at the time of transmission may differ from the algorithm with which the received data is enciphered.
Therefore, there are very few amounts of the data enciphered by the same method.
3. The tapping person cannot encipher the data which he made by the method.
There are some which need biometrics at the time of starting in the latest personal computer, and the data in a hard disk is also usually enciphered.
A tapping person is unable to pour various data enciphered by the communication path by the target method.
4. Determination of the encoding technology to be used can be performed in judgment of an individual.
What is necessary is just to adopt immediately, if it is the outstanding art with which an individual can investigate and trust the contents when new encoding technology is developed.
What is necessary is just to make on one's own, if uneasy. Charge is self-responsibility.
If the encryption software excellent in the individual is purchased, a life of a developer is also secured and a still newer code can be developed.
5. Change to the encoding technology to be used can be performed freely.
The encoding technology which is not reliable is thrown away immediately and can be freely changed into a new thing.
6. A fine response is made to the difference in the law by a country.
Although code mail is receivable for some country and organizations, there can be no regulation

which cannot transmit code mail, only the enciphering key and decryption key which were registered can be used, or only weak encryption is obstructed.
A setup that those who participate in a cryptocommunication network do not do a setup or encryption of encryption finely for every communications partner must be possible.
Since it can set up independently for every mail address the case of transmission, and in the case of reception, cryptocommunication can be used by the method suitable for the law of the country in which the communications partner lives.
Since various algorithms and various keys exist as the whole cryptocommunication network even when submitting a key to a country, even if being submitted is [in it] small and the key flows out outside, the safety as the international whole cryptocommunication network is maintained.

The above matter is big progress in cryptocommunication. I think that it should be recognized as a patent. These became possible because the extended address book had an item of a name, the address, encryption software, an enciphering key, decryption software, and a decryption key.
A firm cryptocommunication network can be built by using the encryption software divided into the encryption portion and the decryption portion in accordance with this.

[Regarding item 27]

Although an answer is the same as 22, describe repeatedly.
It is the art from ancient times which it treats as an attached file, and suitable conversion of base64 etc. is carried out, and is transmitted about this in order to send binary data by E-mail so that an examiner may be said.
It is because he wanted to show that it must be made to have to agree in the existing abbreviation of a communication network since the data which having described this enciphered differs in the data and the property of a basis. Since it was not not much clear, I wanted to emphasize the need by telephone.
Since what cannot transmit is an obvious thing if data does not suit the protocol of a communication network, it may be an item which should be deleted.

[Regarding item 28]

By the method of Leppek, the database for encryption exists, the operation for the encryption used out of it is selected, and in order to determine the sequence which applies it, 160 is generated and used.
By this method, all the people that participate in a cryptocommunication network have to have a common code database.
Who distributes a common database to a new participant how? For some countries, export and import of codes have a strong limit, the person who received the code database becomes an offender only by it, or the person of the way who sent becomes an offender.
If it is the database which collected only extremely weak codes, it may be able to distribute all over the world, but now, it does not become useful.
The retrieving table for pulling out the code operation used from a database itself contains the still bigger problem. Since the portion of the enciphering key to be used does not exist, it is considered to have registered into the retrieving table what had encryption software and an enciphering key.
The length of a safe key becomes still longer. If the length of a key becomes about 4000 bits, a table will overflow from a hard disk.
It is more eager not to use a code database common to the whole.

[Regarding item 29]

Although it is the same as 29.24, it describes repeatedly.
To be sure, as for the sequence of encryption, and the sequence of a decryption, it is obvious that you must be reverse so that an examiner may be said.

For example, if encryption key 51 is specified, the device 14 would first apply encryption algorithm 4, then encryption algorithm 3, then encryption algorithm 1. Conversely, if a message was received by device 14 and the key value 51 was to be used, the device 14 would first decrypt using algorithm 1 then decrypt using algorithm 3 followed by decryption using algorithm 4.

1. A encrypting the data to be sent to B.
2. B decrypting the data from A.
3. B encrypting the data to be sent to A.
4. A decrypting the data from B.

Schier considers 1 and 3 the same way. This will assume the symmetric property of cryptocommunication.
When [however,] Mr. A transmits the data by which multiplex encryption was carried out to Mr. B Some cryptographic algorithms to be used and the cryptographic algorithm used when Mr. A decrypts the data sent by Mr. B need to be completely able to choose another thing, and the structure of an address book where it was extended for realizing it is by no means obvious.
The place which can change the sequence independently applied about the portion of encryption and the portion of a decryption in such an extended address book (in the case of multiplex encryption), respectively serves as a new portion.

[Regarding item 30]

It does not become a telephone, if what was decrypted is not changed into voice so that an examiner may be said.
A difference uses the extended address book. It is in the place considered that the way becomes safer having made separate the method of encryption in case a cipher system in case Mr. A appeals to Mr. B by saying, "How are you?", and Mr. B answer Mr. A "Fine thank you."

[Conclusion]

Believe that the method which the applicant propose shows the art of solving various troubles which a cryptocommunication network has and building an international firm cryptocommunication network.

８．３　決戦延期（２００６年１１月７日）

　最後の決戦だから、全ての力を出し切るぞ！
そう思って、長い長い反論を書きました。私の原案は９０ページでしたが、弁理士さんが整理して４６ページです。１ワード５５円で計算して、１１０万円を超える費用がかかりました。最後の戦いになる。そんな期待を見事に裏切ってくれたのは、これでファイナルだと言ってきたアメリカ特許庁でした。
　アメリカ特許調査様は、継続審査にして再度調査をやり直すか、それとも諦めるかと思ってもいなかった返事を返してきました。
　敵は、消耗戦に切り替えてきたようです。こちらが貧乏で兵糧が切れかかっているのを見越して、戦法を切り替えてきたようです。問題はお金です。
　何とかしてお金を工面しなくてはなりません。未だ経済産業省からの返事はないし、ヨーロッパからの正式書類は届いていない現状では、城の中に畑を作って食べ物を育てるしか有りません。そうです。他のソフトを売って稼ぐしか有りません。幸いウェーブレットのソフトがお米を持ってきてくれます。

弁理士さんからのメールです。

宇山様
お世話になっております。
さて、標記の件ですが、数日前に、米国代理人より再度の庁指令が届きました。その内容によりますと、補正書により参照テーブルの内容を明確化したため、再度の調査が必要となり、現時点では、出願人に対して、（１）継続出願に切り替えて、調査をやり直す、（２）審判請求を行い、上級審に進む、（３）なにもせずに、権利化をあきらめる、の選択を求めております。
このような場合は、通常、継続出願に切り替えるのが普通です。ただし、問題は、米国における費用負担がどれくらいになるか、です。ご承知のように、前回の反論書提出も含めて、私の費用は極力抑えておりますが、米国代理人費用は私ではコントロールできません。
そこで、米国代人に、前回の反論書提出の費用、今後の継続出願の費用を至急問い合わせてみますので、宇山さんには、その請求額をご覧になってから、決定をされてはどうかと思います。今しばらく、お待ち下さい。

こんなアメリカ特許庁ですが、問題はお金だけです。稼げば済むことです。多くの企業や大学がウェーブレットのソフトを買うという形で私の見方をしてくれています。ここで白旗を掲げるわけには行きません。
私はいつも、畑のアブラムシと戦っています。アブラムシは命がけで私の撒く農薬と戦います。アメリカ特許庁は単なる兵糧責めをしてくるだけです。少しはアブラムシの勇気を見習ったらどうですか。
アブラムシと比べたら、へなちょこ、へっぽこアメリカ特許庁ですから、最後に白旗を掲げるのはアメリカ特許庁に決まっています。

８．４　勝利の旗は閃かん（２００７年１月１７日）

アメリカの代理人から連絡が入りました。“請求項に関して簡単な修正を受け入れるならば特許認定をすると審査官が電話してきた。”とのことでした。気の弱い私はすぐに了解しました。
何でも受け入れます。認めてください。

８．５　許可通知（２００７年３月１日）

ついに、特許許可通知がでたのです。
次の日は卒業式の日でした。卒業してゆく生徒たちにアメリカで特許が取れたと報告が出来ました。
今年は暖冬で春の来るのが早く、朝から良く晴れて風も穏やかな一日でした。

８．６　特許番号決定

やっと、アメリカでの特許番号が決まりました。
U.S. Patent　US 7219229
です。

９．暗号の輸出規制

９．１　法律の規制

令第９項

（７）
暗号装置又はその部分品
貨物等省令第８条

１項　九　号

(第８条第九号：告示貨物)

暗号装置であって、次のイから二までのいずれかに該当するもの（次のホからヌまでのいずれか又は第三条

第十九号ハ（二）２又は第十条第五号イに該当するものを除く。）又はその部分品

イ　デジタル方式の暗号処理技術（アナログ方式の暗号処理をデジタル方式の暗号処理技術を用いて実行するものを含む。）を用い、認証又はデジタル署名のため以外の暗号機能を有するように設計したものであって、次のいずれかに該当するもの

（一）　対称アルゴリズムを用いたものであって、アルゴリズムの鍵の長さが５６ビットを超えるもの

（二）　非対称アルゴリズムを用いたものであって、アルゴリズムの安全性が次のいずれかの有する困難性に基づくもの

１　５１２ビットを超える整数の素因数分解

２　有限体上の乗法群における５１２ビットを超える離散対数の計算

３　２に規定するもの以外の群における１１２ビットを超える離散対数の計算

ロ　暗号解析を行うように設計したもの

ハ　スペクトル拡散のための拡散符号の生成（周波数ホッピングのためのホッピング符号の生成を含む。）に暗号処理技術を用いるように設計したもの

ニ　ウルトラワイドバンド技術のためのチャンネル符号又はスクランブル符号の生成に暗号処理技術を用いるように設計したもの

ホ　ＩＣカードであって、その暗号機能が次のへからヌまでのいずれかに該当する装置に限定されて使用されるもの 又は広く一般に使用されるものであって、その暗号機能が使用者によって変更できず、かつ、当該 IC に保存されている個人情報の保護に限定して 使用するように設計したもの

ヘ　ラジオ放送又は特定加入者用放送（テレビジョン放送にあっては、有線テレビジョン放送を含む。）の受信装置であって、復号化機能のみ又は加入者から放送事業者（テレビジョン放送事業者にあっては、有線テレビジョン放送事業者を含む。）への課金情報若しくは番組関連情報を送信するための暗号化機能のみを有するもの

ト　使用者によって暗号機能の変更ができない装置であって、暗号機能として次のいずれかに該当する一又は二以上のもののみを有するように設計したもの

（一）　コピー防止されたプログラムを実行するための機能　　（二）　コピー防止された読み出し専用媒体上のデジタルコンテンツにアクセスをするための機能

（三）　同一内容で公衆に販売される媒体上に暗号化して記憶されたデジタルコンテンツにアクセスするための機能

（四）　著作権が保護された音声又は映像データの複製を管理する機能

チ　暗号装置であって、銀行業務又は金融決済業務に使用するように設計したもの

リ　民生用の携帯用電話機端末又は移動用電話機端末であって、電話機端末間での暗号化機能を有しないもの

ヌ　コードレス電話機端末間での暗号化機能を有しないコードレス電話装置であって、コードレス電話機端末と家庭内基地局の間に無線中継器がない場合の一無線区間での電波到達最長実効距離が４００メートル未満のもの

－－±－－　　－－－　　－－－－

この方法はもちろん電話でも可能です。ただし、電話機の場合は法律の規制が厳いので、作って売るのは難しそうです。IP 電話は可能性があります。パケット通信の規制はゆるいのです。これは、携帯電話で世界一のノキアに問い合わせをして教えていただいたことです。

この中で示された方式で暗号通信を行う携帯電話は作れても売れないが、IP 電話のソフトなら違法性なく制作し日本国内では使用できる。ただし、輸出には制限が付く。

この制限が、ソフトのソースコードを完全に公開する事で消えてしまうのか否か、経済産業省の相談しなくてはなりません。でも、ソースコードを公開したソフトを買ってくれる人はいるのだろうか？特許は著作権とは違うので、ソースコードを改変したくらいでは、著作権違反は防げても、特許権侵害は防げない。そう考えています。

正義の国アメリカにも、盗聴をする悪者がいるらしい。アメリカで特許がとれれば、ソースコードを公開してもアメリカでそのソースコードを見てソフトを作って販売する人から、特許使用料は取れると思っています。ただし、技術的な質問に個別的には答えられないので、誰でもすぐにコンパイル出来るように丁寧に解説したソースコードにしなくてはならない。なぜなら、経済産業省様は次のようにおっしゃられます。

【大学・研究機関の皆様へ】
　安全保障貿易管理の目的は、前述のとおり大量破壊兵器の拡散防止等を目的としており、大学・研究機関における研究開発を阻害することを意図している訳ではありません。このため、不特定多数の者に対する技術の提供（公知の技術提供）等は、当該許可を免除する特例規定があります。
　また、海外出張等の際、個人的な使用目的でハンドキャリーされるパソコン等については、たとえ規制対象貨物であっても許可を取得する必要はありません。しかし、これを海外において他人に使用させたり、ソフトを譲渡する場合等は技術提供に係る許可が必要となる場合があります。
＜許可が免除される「公知の技術提供」の例＞
○学会誌、公開特許情報、公開シンポジウムの議事録等、不特定多数の者が入手可能な技術の提供
○講演会、展示会等において不特定多数の者が入手又は聴講可能な技術の提供
○学会発表用の原稿又は展示会等での配布資料の送付、雑誌への投稿等
＜許可が免除される「貨物の輸出」の例＞
○海外出張等の際、一時的に海外にパソコンを持ち出して個人的に使用する場合
○輸出される貨物の金額が少額の場合（貨物や輸出先によって異なりますのでご注意下さい。）
　当然のことながら、上述の特例規定が適用されない場合については、個人や大学・研究機関等であっても輸出等を行う場合には許可申請が必要です。特に、以下のような場合には、安全保障貿易管理に係る許可申請の必要性を念頭においていただき、かかる貨物等が規制対象貨物等であるか否かを十分に精査いただいた上で、許可が必要な場合には事前に経済産業省までご相談・許可申請してください。
＜注意すべき「技術の提供」の例＞
○海外の共同研究先等又は個別の研究者への技術資料・プログラム等の提供
○日本の研究機関での研究員、留学生、研修生等の受入れに伴う技術の提供
○海外の研究機関に対する特許使用許諾に伴うノウハウ等の提供
○日本の研究機関における研究室の見学時の技術の提供
＜注意すべき「貨物の輸出」の例＞
○海外の研究機関への装置等の送付
○海外で開催される学会・シンポジウム等への装置等の出品
○海外の共同研究先又は個別の研究者への試料・試作品等の送付、装置等の貸与
○海外出張時に手荷物として持ち出す試料・部品・試作品・測定機器等
【お問い合わせ先】
本件に係るお問い合わせについては、
経済産業省　貿易経済協力局　安全保障貿易検査官室　０３－３５０１－２８４１
までお願いします。

特に、

＜注意すべき「技術の提供」の例＞
○海外の共同研究先等又は個別の研究者への技術資料・プログラム等の提供

から、個別の人に教えてはいけない。従って技術的な質問には答えない。答える必要がないくらい親切なソースコードを公開する必要がある。

また、

＜許可が免除される「公知の技術提供」の例＞
○学会誌、公開特許情報、公開シンポジウムの議事録等、不特定多数の者が入手可能な技術の提供

から、誰でも見ることが出来るホームページや、本にして出版するのは自由らしい。

さらに、

＜許可が免除される「貨物の輸出」の例＞
○海外出張等の際、一時的に海外にパソコンを持ち出して個人的に使用する場合

から、日本人なら、このソフトを搭載したパソコンを一時的に海外で使用できる。

　何とか使えそうですね。

これを書いたり、楕円曲線暗号の本を読んだり、メーラーの改良をしたり、IP 電話のソフトを考えたり、と私は大忙しです。
泣きそうです。

アメリカの暗号規制について、郵政省の調査研究によれば、
(2)米国
米国はこれまで、段階的に暗号製品の輸出規制を緩和してきている。
平成 8 年 12 月に、鍵長 40 ビット以上 56 ビット未満の暗号製品には、鍵回復（key recovery）機能を搭載する計画を示すこと及び、56 ビット以上の暗号製品は、輸出の際に許可を得ることを条件に、それまでの 40 ビットより強力な暗号製品の輸出を許可した。また、この時、暗号輸出規制の所管が国務省から商務省に移管された。
次いで、平成 10 年 7 月には、指定 45 ヶ国の金融機関が電子商取引に暗号製品を利用する場合、鍵長や鍵回復機能の有無にかかわらず輸出を許可した。そして、同年 9 月には、7 月の緩和策を保険会社、医療機関、オンライン通販会社に拡大すると同時に、56 ビット未満の暗号製品の輸出に必要としていた条件を撤廃した。
平成 12 年 1 月、米国商務省は、暗号製品の輸出に関する規制緩和を発表した。この規制の主な概要は、以下の通りである。
テロ支援 7 カ国（キューバ、イラン、イラク、リビア、北朝鮮、スーダン、シリア）を除く非政府機関のいかなる消費者に対して、すべての暗号製品は、政府の許可なしに輸出することができる。ただし、技術審査は必要である。「小売暗号商品及びソフトウエア」は、テロ支援 7 カ国を除き、いかなる消費者に対しても輸出することができる。米国企業の海外子会社へは、技術審査することなく、暗号製品を輸出することができる。非米国企業等に鍵長 64 ビットを超える暗号製品を輸出した場合、報告する必要がある。ただし、金融特化型製品、個々の消費者※を対象とする製品、ダウンロードなどを通じて入手できる製品などには報告は不要である。

さらに、暗号の輸入規制は無いようです。

日本の法律さえクリアすれば、輸出ではなくて、勝手にソフトを作ってアメリカで販売しているアメリカ人から特許使用料を取れそうです。
経済産業省様に伺う必要がありますが、質問が来たらその答えを、ホームページに公開してしまえば良さそうです。

９．２　経済産業省へのお願い（２００６年８月２９日）

２００６年８月２９日東京の経済産業省へ行きました。前の日の電話したら相談は予約できない早く来たものから順に並ぶ。相談の開始は１０時から、受付は９時からとのことでした。
朝６時に出発して、電車に１時間くらい乗って、新橋についてから２０分くらい歩いてやっと、朝８時頃に経済産業省に着きました。入り口にガードマンがいて身分証明書の提示を求められました。免許証を持っていたので通してもらえました。次に手荷物検査があります。鞄を開いて中の資料を見せました。
エレベーターが見つからなくてうろうろしていました。反対側の出口にガードマンさんがいましたのでやっとエレベータの場所が分かり無事に１４階につきました。相談する所には誰もいません。廊下にイスと小さな机があったので座って待っていました。１４階からの眺めはなかなかのものでした。もちろん私が１番乗りでした。慣れている会社の人は９時頃に来ていました。９時になって待合室が開きました。中には３０人分くらいのイスがありました。待合室はクーラーが入って快適でした。
一番で相談申し込みの書類を書きました。間違えました。暗号関係は貨物（暗号電話など）と役務（暗号ソフトなど）があります。ソフトの相談なので役務なのに、申込用紙の左側貨物の欄に書いてしまいました。係

りの人と話したらどうも変です。原因は私の記入ミスでした。ごめんなさい。しばらく待って役務の人が来ました。少し相談をしたら、その人はいったん奥に引っ込んで１０分くらい出てきませんでした。出てきた後はいきなり歯切れが悪くなり、相談が進みません。結局資料を送ってくださいとのことでした。
一番の理由は、言った言わないの様な事になると困るので、資料を送ってくれたら文書で答えてくれる。とのことでしたので、一応納得して、送る事にしました。日本語は短い簡単なものを、英語は長い面倒なものを送ることにしました。しかし、これでソフトの性質を判断出来るのだろうか？なにしろ、作った本人が良く分からないのですから。

仕方ないので、ソースコードの公開が許されるか否かを電話で確認して、さらに沢山のデータをホームページで公開してから資料を送ることにしました。
電話では、以下の記述の意味を聞きました。

＜許可が免除される「公知の技術提供」の例＞
〇学会誌、公開特許情報、公開シンポジウムの議事録等、不特定多数の者が入手可能な技術の提供
〇講演会、展示会等において不特定多数の者が入手又は聴講可能な技術の提供
〇学会発表用の原稿又は展示会等での配布資料の送付、雑誌への投稿等
＜許可が免除される「貨物の輸出」の例＞
〇海外出張等の際、一時的に海外にパソコンを持ち出して個人的に使用する場合

制限なしに誰でも入手できる技術の提供にホームページでのソースコードの公開が含まれるとのことでした。雑誌も可能です。どんどん書くぞ。
Protected Communication System　は暗号ソフトではない！
今はそう思っています。昔は勘違いしていて、世界最強の暗号ソフト、暗号通信システムだと思っていました。ごめんなさい。これは単なるバッチファイルのようなものです。
だから、輸出を許可してください。もしだめなら、ソースコードを公開して公知の技術にいたします。たしか、そのときは許可なしで海外に送れるのですよね。出版社の皆さん協力して下さいね。
ソースコードを書いた本が出るときは、私は儲からないと言うことです。原稿料か印税で我慢します。

さて、その経済産業省へ提出したお願いの文書は以下のものです。内容は単なるへりくつの固まりです。でも、一理ある。自分が受け取ったらいやになる文書です。２００６年９月１２日、午後６時頃に郵便局で配達記録郵便で送りました。

－－－－－－－－－－－－－　ここから　－－－－－－－－－－－－－－－－

Protected communication system について

宇山靖政

役務担当　　＊＊　様

８月２９日に伺ったときに、資料を送れば検討していただけることになりました。特許申請関係の資料として、欧州特許庁のホームページから公開されているもの、ＣＱ出版社の雑誌に掲載されたもの、インターネットで自由に見ることが出来るマニュアルの３点を送ります。特許取得を目指したので記述内容が暗号に偏っています。このソフトの性質を客観的な立場から説明しておきたいと思います。

１．このソフトの名前は Protected communication system です。機能はメールの送信と受信です。
２．さらに特別な機能として、別のソフトを登録できます。
３．別のソフトが登録されていない場合は、普通のメールソフトと同様に電子メールの送受信ができます。
４．別のソフトが登録されていた場合には、
　４－１．送信時には登録されているソフトを呼び出して、送信するデータを、その呼び出されたソフトに渡します。その登録されたソフトが返してきたデータを送信します。
　４－２．受信時には登録されているソフトを呼び出して、受信したデータを、その呼び出されたソフトに渡します。その登録されたソフトが返してきたデータを表示します。
５．登録するソフトに特に制限はありません。どんなソフトでも呼び出せます。
　例えば、圧縮、解凍ソフトが登録されていれば、送信時には、送信するデータを圧縮ソフトに渡して、圧縮されたデータが帰ってきたら、それを送信します。受信時には、受信したデータを解凍ソフトに渡します。解凍されて帰ってきたデータを表示します。
　暗号化ソフトと復号化ソフトを登録すれば暗号化されたデータが送信され受信され

たものを復号化してから表示することになります。
6．このソフト（Protected communication system）の機能は、
他のソフトが登録されていなければ、普通の電子メール送受信用のソフトであり、
他のソフトが登録されていれば、登録されたソフトにデータを渡し、帰ってきたデータを送信する。受信したデータを登録されたソフトに渡し、帰ってきたデータを表示する。というような機能です。
7．このソフト単独では暗号化機能を持っていません。
しかも、他のソフトを呼び出しデータの受け渡しをするという機能は変更できません。
【質問事項】
8．気になるのは、このソフト（Protected communication system）は、
”暗号機能等プログラム”に相当するか？　それとも暗号機能を持っていないソフトか？
”暗号機能が使用者によって変更出来ないもの”に相当するか？
このソフト自体の機能、どんなソフトでも登録できて、登録されたソフトとの間でデータの受け渡しをするという機能は変更できません。
9．現在、ＣＱ出版社の雑誌への投稿。日本、米国、欧州への特許申請をしておりますが、この事実だけで、このソフトは、”プログラムが公知の技術になっているもの”と認められるのでしょうか？　投稿内容や特許申請の内容だけでは、公知とは認められないならば、公知になるための条件について教えて下さい。
ソースコードをホームページで不特定多数の人が入手可能な形で公開する。
ソースコードを雑誌に投稿する。
ソースコードを解説した本を出版する。
ソースコードを明細部に記載した特許申請をする。
など、がその条件となるのではないかと考えていますが、いかがでしょうか？
10. 現在はヨーロッパでの特許取得に成功し、フィンランド、ドイツ、イギリスでの特許登録をする予定です。アメリカの結果は１２月頃に判明する予定です。日本では具体的な技術ではないという理由で認定されませんでした。このソフトを特許取得した国に輸出することは可能でしょうか？

なお、１２月には米国特許庁の結論が出ますので、その頃までに御返事をいただければ幸いです。

－－－－－－－－－－－－　ここまで　－－－－－－－－－－－－－－－－－

もちろん、出版しても良いことや、ＨＰで公開しても良いことは、これを書く前に経済産業省の人に電話で確認してあります。
どんな返事が来るのか楽しみに待っています。

９．３　経済産業省の返事（２００６年１２月１４日）

やっと返事が来ました。でも文書ではなかったです。内容は
1．メーラーは単独では暗号機能を持っていないので販売には問題なし。
2．暗号ソフトはソースコードをホームページで公開すれば、許可無く配布できる。

当然といえば当然の答えですが、やはり私のメーラーは暗号ソフトではなかったようです。
たとえば、トラックに爆弾を積めば全体としては武器になりますが、トラックだけでは武器にはなりません。トラックにはいろいろなものを積むことが出来ます。
私のソフトも、暗号ソフトと組み合わせることも出来ますが、圧縮ソフトと組み合わせることも出来ます。従ってトラック自体が武器に当たらないのと同じですね。
ウイニーの例もあるのですが、あれは著作権法に違反する行為を助長するとして有罪でしたが、私のソフトは個人情報を保護するのだから問題なしというところでしょうか。

新しい暗号は美しく楽しい暗号にします。

９．４　特許実施許諾（２００７年６月１１日）

輸出と海外での特許実施許諾の関係が不明だったので、再度、経済産業省に問い合わせました。

アメリカ、ドイツ、イギリス、フィンランドでの特許が確定し、最後の難関、日本の法律ですが、これも、クリアーしてしまったようです。

以下、その経過です。

> 特許実施許諾と輸出許可の関係について質問します。
>
>
> 宇山靖政
>
> 電子メール： uyama@kc4.so-net.ne.jp
>
> ５月２８日に電話で質問したところ、FAX で送信してくれとのことでしたので
> 簡単にまとめました。
>
> 以前、西沢さん宛てに、資料１と特許申請した内容、HP に掲載してある内容、
> 特許の解説を書いた雑誌の記事の三点を送りました。そのころは、特許も確定
> していない状態で、暗号やメーラーのソフト開発を進めていました。西沢さん
> からは、内容を HP で公開すればよいとのアドバイスをいただきました。現在は
> 暗号ソフトのソースコード６種類ほどを公開しています。さらに追加する予定
> です。
>
> このたび、アメリカ、ヨーロッパ（イギリス、ドイツ、フィンランド）にて
> 特許を取得いたしました。
>
> 特許番号：<U.S. Patent US7219229 および European Patent EP1244267>となっています。各国の特許庁の HP などで閲覧できます。（Protected communication system で検索）
>
> 特許の内容は、暗号メールと暗号電話に関するものです。特許に記載されている内容では
> １．利用者が自由に暗号ソフトを変更できる。
> ２．暗号化鍵の長さはユーザーが必要に応じて長くできる
> となっています。このことから、日本で作成した暗号電話機や暗号メールソフトを外国に輸出するのは、経済産業大臣の許可が必要と考えています。この認識は正しいでしょうか（質問１）
>
> 取得した特許に関して、外国の企業が電話機として、作成し販売することを認めて（特許実施許諾を与えて）作成、販売してもらうには経済産業大臣の許可が必要でしょうか？あるいは、他の許可や手続きは必要でしょうか？（質問２）
>
> 取得した特許に関して、外国の企業がソフトウエアとして、作成し販売することを認めて（特許実施許諾を与えて）作成、販売してもらうには経済産業大臣の許可が必要でしょうか？あるいは、他の許可や手続きが必要でしょうか？（質問３）
>
> 契約をするとき、
> 公開された公開された特許の内容以上の技術的な質問に答えてはいけないのでしょうか？（質問４）
>
> 個別の質問があったとき、回答は HP に公開して、質問者は HP を見るようにする。もちろん他の人も自由に見ることができるようにしてあるならば、質問に HP を使って答えても良いのでしょうか？（質問５）
>
> 質問に答えることができないなら、質問されそうな事柄を予測し、事前に HP で技術情報の詳細を公開しておくことは問題無いでしょうか？（質問６）
>
> 経済産業省で相談するとき、申し込みは役務と貨物のどちらにすれば良いのでしょうか？（質問７）

>
>
>　電子メールで返事がもらえるとありがたいのですが、電話でもかまいません。お忙しいところ申し訳ありませんがよろしくお願いします。
> ２８日に電話する前は、輸出と特許実施許諾を同じように考えていて、さらに、輸出許可は下りないと思っていましたので、世界中で実施してもらうには特許権と著作権を放棄する（無料とする）必要があるのかと思っていました。
>

> 資料１：　　Protected communication system について
>
> 宇山靖政
>
> 役務担当　　西沢　様
>
>　８月２９日に伺ったときに、資料を送れば検討していただけることになりました。特許申請関係の資料として、欧州特許庁のホームページから公開されているもの、ＣＱ出版社の雑誌に掲載されたもの、インターネットで自由に見ることが出来るマニュアルの３点を送ります。特許取得を目指したので記述内容が暗号に偏っています。このソフトの性質を客観的な立場から説明しておきたいと思います。
>
> １．このソフトの名前は Protected communication system です。機能はメールの送信と受信です。
>
> ２．さらに特別な機能として、別のソフトを登録できます。
>
> ３．別のソフトが登録されていない場合は、普通のメールソフトと同様に電子メールの送受信ができます。
>
> ４．別のソフトが登録されていた場合には、
>
>　４－１．送信時には登録されているソフトを呼び出して、送信するデータを、その呼び出されたソフトに
>
>　　渡します。その登録されたソフトが返してきたデータを送信します。
>
>　４－２．受信時には登録されているソフトを呼び出して、受信したデータを、その呼び出されたソフトに
>
>　　渡します。その登録されたソフトが返してきたデータを表示します。
>
> ５．登録するソフトに特に制限はありません。どんなソフトでも呼び出せます。
>
>　例えば、圧縮、解凍ソフトが登録されていれば、送信時には、送信するデータを圧縮
>
>　ソフトに渡して、圧縮されたデータが帰ってきたら、それを送信します。受信時には、
>
>　受信したデータを解凍ソフトに渡します。解凍されて帰ってきたデータを表示します。
>
>　　暗号化ソフトと復号化ソフトを登録すれば暗号化されたデータが送信さ

> れ受信され
>
>　　たものを復号化してから表示することになります。
>
> ６．このソフト（Protected communication system）の機能は、
>
>　　他のソフトが登録されていなければ、普通の電子メール送受信用のソフト
> であり、
>
>　　他のソフトが登録されていれば、登録されたソフトにデータを渡し、帰っ
> てきたデー
>
>　　タを送信する。受信したデータを登録されたソフトに渡し、帰ってきた
> データを表示
>
>　　する。というような機能です。
>
> ７．このソフト単独では暗号化機能を持っていません。
>
>　　しかも、他のソフトを呼び出しデータの受け渡しをするという機能は変更
> できません。
>
> 質問事項
>
> ８．気になるのは、このソフト（Protected communication system）は、
>
>　”暗号機能等プログラム” に相当するか？　それとも暗号機能を持っていな
> いソフトか？
>
>　”暗号機能が使用者によって変更出来ないもの”に相当するか？
>
>　このソフト自体の機能、どんなソフトでも登録できて、登録されたソフトと
> の間でデータの受け渡しをす
>
>　るという機能は変更できません。
>
> ９．現在、ＣＱ出版社の雑誌への投稿。日本、米国、欧州への特許申請をして
> おりますが、この事実だけで、
>
>　このソフトは、”プログラムが公知の技術になっているもの”と認められるの
> でしょうか？　投稿内容や特
>
>　許申請の内容だけでは、公知とは認められないならば、公知になるための条
> 件について教えて下さい。
>
>　ソースコードをホームページで不特定多数の人が入手可能な形で公開する。
>
>　ソースコードを雑誌に投稿する。
>
>　ソースコードを解説した本を出版する。
>
>　ソースコードを明細部に記載した特許申請をする。
>
>　など、がその条件となるのではないかと考えていますが、いかがでしょうか？
>

> 10. 現在はヨーロッパでの特許取得に成功し、フィンランド、ドイツ、イギリ
> スでの特許登録をする予定で
>
>　す。アメリカの結果は１２月頃に判明する予定です。日本では具体的な技術
> ではないという理由で認定さ
>
>　れませんでした。このソフトを特許取得した国に輸出することは可能でしょ
> うか？
>
> なお、１２月には米国特許庁の結論が出ますので、その頃までに御返事をいた
> だければ幸いです。
>

経済産業省からの返事は、

> 宇山　様
>
> お世話になります。
>
> 西沢の件、当方の手違いで引き継ぎがなく申し訳ありませんでした。
>
> 電話にてお話申し上げましたが、今後のこともあるので
> メールさせていただきます。
>
>
> まず、今回の件、暗号機能を有するソフトを提供する際は、輸出貿易管理令
> 別表第１の８項ないし９項の規制に従い、それらを実現するプログラムとして
>
> 役務取引許可の対象となります。
>
> ただし、特例があって、市販品つまりパッケージソフトのように不特定多数が
> 購入ないし入手が可能で、かつ、暗号機能が使用者によって変更されない、
>
> かつ、使用者への技術的支援も不要なものとされている場合、許可を要しない
> こととしています。
>
> また、一方で今回の件、それを作成、設計技術情報が、公開特許情報になって
> いるため、公知の技術として役務取引許可不要となります。
>
> 本来、外為法でいうところの役務取引許可が必要かどうかの判断は、これを
> 海外の特許事務所なり弁理士等に提供する時点で行われるべきであり、
>
> その時点が役務取引にあたります。（技術を公知としようとすることで許可
> 不要という判断になるとは思いますが…。）
>
> したがって公開された特許情報をもって、許諾を与えるかどうかはすでに外為法
> の範疇ではなくなっています。各国の特許法制に従っていただくということでしょ
> うか。
>
>
> 以上のことから、質問１については正しいです。質問２、３は外為法の範疇外
> 質問４は貿易外省令で不要とした役務提供の範囲を超えるので、許可要です。
> 質問５、６は質疑も含め、公開してしまうのであれば、許可不要です。
>

> つまり、今回の件も外国で出願する時点が役務取引であって、それが済んで
> しまっている状態で相談を受けても、もし公知で読めない場合、すでに違反が成立
> して
>
> しまうわけです。ソフト（プログラム）を御者が自ら、海外の人（非居住者）に提
> 供
> （提供地は問いません。国内であっても）することになるような場合は是非事前に
>
> ご相談ください。
>
> **********
> *******
> 経済産業省貿易経済協力局貿易管理部
> 安全保障貿易審査課
>
> TEL03-3501-2801/fax03-3501-6004
> 〒100-8901 千代田区霞が関１－３－１
>

となりました。

したがって、外国での特許実施許諾を与えることが可能になってしまいました。
もし、上記の国にお住まいの方で、プログラムや電話を作成することに協力してくれそうな方や企業の情報がありましたら教えていただければ幸いです。

９．５　お金を下さい

経費は次の表のようになっています。もちろん、ソフトの開発費用は含まれません。
ここには含まれていませんが、さらにアメリカ特許庁の分がよけいにかかります。

年度	合計
2001 H13	¥1,281,040
2002 H14	¥1,903,758
2003 H15	¥348,386
2004 H16	¥444,742
2005 H17	¥754,329
2006 H18	¥2,979,904
合計金額	¥7,712,159

このほかに、いま分かっているだけで、アメリカの再調査関係で７０万円、ドイツ語とフィンランドの翻訳費用、特許維持費用などがかかりますので、さらに３００万円くらいは用意しないといけないと思っています。

計画では、アメリカ、イギリス、ドイツ、フィンランド、日本の５カ国で取るつもりだったのですが、日本は取れませんでした。ヨーロッパは取れましたが、事務手続きが残っています。

アメリカにはだまされました。最終反論だといわれてその気になった私がばかでした。なんと継続審査になるとのことでした。こうなると、振り込め詐欺と変わらない気がします。一度だまされたら次々にだましの集

団が押し寄せる。

　費用の回収ですが、
Ⅰ．メーラーは一部機能制限を付けてインターネットで配布する。対象は全世界としお金をもらったら機能制限解除の鍵を送る。

Ⅱ．暗号ソフトは、
（1）他でもソースコードが公開されているものはソースコードを公開し、自分でコンパイルできる人は勝手に使えるようにする。コンパイルできない人に対してはインターネットを使って販売する。
（2）ソースコードを公開できないものは、日本国内のみ郵送で販売する。

それにしても、１０００万円を回収するのは大変そうです。

//

１０．暗号ソフトのソースコード

　暗号ソフトはソースコードを公開してしまえば、公知の技術となって自由に使えるようになります。共通鍵方式の暗号ソフトで高性能のもののソースコードがいくつか公開されています。それらは、自分のソフトに組み込んで使うには修正が必要です。
　私は、Windows 上の 32 ビット CPU（Pentium など）で動くコンソールタイプの暗号ソフトに書き換えました。一部分本来のものと変更している部分があります。
　使用したコンパイラはＶＣ++6.0 です。ここに書かれたソースコードを使えば皆様自身で必要なソフトを作成できると思います。法律の制限で個別の質問にはお答えできませんが、問題点などは HP に記載する形で解決してゆきたいと思っています。
　この Protected Communication System(Mail-J) に付属する暗号ソフトのうち、本来のソースコードが公開されている、Camellia, AES, Twofish, Serpent, MARS を利用したものについて、本来公開されているソースコードの一部を私が変更したものを公開いたします。
　私の変更に関して安全性に不安のある方は、私のホームページから本来のソースコードへのリンクが張ってありますので、それらを入手されて、本来のものと変更したものの両方のソースコードを比較されて、ご自分で信頼できるものを作成し、御自分でコンパイルされて使用されることをお勧めします。私が使用したコンパイラは VC++ 6.0 です。
　著作権に関しては私が所有する部分があります。変更は本来のソースコードを基にして行なってください。MARS 暗号に関してだけは私が変更したものを元にして変更してもかまいません。

１０．１　Camellia 暗号ソースコード

　掲載したソースコードの大部分はＮＴＴと三菱電機様が公開されたものですが、このメールシステムで利用するために少しだけ変更してあります。
　安全性に関しては皆様が本来の Camellia 暗号のソースコードと比較されて確認してくださることを希望いたします。私のホームページからリンクが張ってありますのですぐに見つかると思います。

１０．１．１　暗号化部分

```
/*****************************************************
 *                                                   *
 *        Camellia Block Encryption Algorithm        *
 *          in ANSI-C Language : Camellia.c          *
 *                                                   *
 *          Version M1.02 September 24 2001          *
 *   Copyright Mitsubishi Electric Corp 2000-2001    *
 *                                                   *
 *****************************************************/
#include "stdafx.h"
```

```
typedef unsigned char Byte;
typedef unsigned long Word;

void Camellia_Ekeygen( const int, const Byte *, Byte * );

void Camellia_Encrypt( const int, const Byte *, const Byte *, Byte * );
void Camellia_Decrypt( const int, const Byte *, const Byte *, Byte * );

void Camellia_Feistel( const Byte *, const Byte *, Byte * );
void Camellia_FLlayer( Byte *, const Byte *, const Byte * );

void ByteWord( const Byte *, Word * );
void WordByte( const Word *, Byte * );
void XorBlock( const Byte *, const Byte *, Byte * );
void SwapHalf( Byte * );
void RotBlock( const Word *, const int, Word * );

const Byte SIGMA[48] = {
0xa0,0x9e,0x66,0x7f,0x3b,0xcc,0x90,0x8b,
0xb6,0x7a,0xe8,0x58,0x4c,0xaa,0x73,0xb2,
0xc6,0xef,0x37,0x2f,0xe9,0x4f,0x82,0xbe,
0x54,0xff,0x53,0xa5,0xf1,0xd3,0x6f,0x1c,
0x10,0xe5,0x27,0xfa,0xde,0x68,0x2d,0x1d,
0xb0,0x56,0x88,0xc2,0xb3,0xe6,0xc1,0xfd};

const int KSFT1[26] = {
0,64,0,64,15,79,15,79,30,94,45,109,45,124,60,124,77,13,
94,30,94,30,111,47,111,47 };
const int KIDX1[26] = {
0,0,4,4,0,0,4,4,4,4,0,0,4,0,4,4,0,0,0,0,4,4,0,0,4,4 };
const int KSFT2[34] = {
0,64,0,64,15,79,15,79,30,94,30,94,45,109,45,109,60,124,
60,124,60,124,77,13,77,13,94,30,94,30,111,47,111,47 };
const int KIDX2[34] = {
0,0,12,12,8,8,4,4,8,8,12,12,0,0,4,4,0,0,8,8,12,12,
0,0,4,4,8,8,4,4,0,0,12,12 };

const Byte SBOX[256] = {
112,130, 44,236,179, 39,192,229,228,133, 87, 53,234, 12,174, 65,
 35,239,107,147, 69, 25,165, 33,237, 14, 79, 78, 29,101,146,189,
134,184,175,143,124,235, 31,206, 62, 48,220, 95, 94,197, 11, 26,
166,225, 57,202,213, 71, 93, 61,217,  1, 90,214, 81, 86,108, 77,
139, 13,154,102,251,204,176, 45,116, 18, 43, 32,240,177,132,153,
223, 76,203,194, 52,126,118,  5,109,183,169, 49,209, 23,  4,215,
 20, 88, 58, 97,222, 27, 17, 28, 50, 15,156, 22, 83, 24,242, 34,
254, 68,207,178,195,181,122,145, 36,  8,232,168, 96,252,105, 80,
170,208,160,125,161,137, 98,151, 84, 91, 30,149,224,255,100,210,
 16,196,  0, 72,163,247,117,219,138,  3,230,218,  9, 63,221,148,
135, 92,131,  2,205, 74,144, 51,115,103,246,243,157,127,191,226,
 82,155,216, 38,200, 55,198, 59,129,150,111, 75, 19,190, 99, 46,
233,121,167,140,159,110,188,142, 41,245,249,182, 47,253,180, 89,
120,152,  6,106,231, 70,113,186,212, 37,171, 66,136,162,141,250,
114,  7,185, 85,248,238,172, 10, 54, 73, 42,104, 60, 56,241,164,
 64, 40,211,123,187,201, 67,193, 21,227,173,244,119,199,128,158};

#define SBOX1(n) SBOX[(n)]
#define SBOX2(n) (Byte)((SBOX[(n)]>>7^SBOX[(n)]<<1)&0xff)
#define SBOX3(n) (Byte)((SBOX[(n)]>>1^SBOX[(n)]<<7)&0xff)
#define SBOX4(n) SBOX[((n)<<1^(n)>>7)&0xff]
```

```
void Camellia_Ekeygen( const int n, const Byte *k, Byte *e )
{
	Byte t[64];
	Word u[20];
	int  i;

	if( n == 128 ){
		for( i=0 ; i<16; i++ ) t[i] = k[i];
		for( i=16; i<32; i++ ) t[i] = 0;
	}
	else if( n == 192 ){
		for( i=0 ; i<24; i++ ) t[i] = k[i];
		for( i=24; i<32; i++ ) t[i] = k[i-8]^0xff;
	}
	else if( n == 256 ){
		for( i=0 ; i<32; i++ ) t[i] = k[i];
	}

	XorBlock( t+0, t+16, t+32 );

	Camellia_Feistel( t+32, SIGMA+0, t+40 );
	Camellia_Feistel( t+40, SIGMA+8, t+32 );

	XorBlock( t+32, t+0, t+32 );

	Camellia_Feistel( t+32, SIGMA+16, t+40 );
	Camellia_Feistel( t+40, SIGMA+24, t+32 );

	ByteWord( t+0,  u+0 );
	ByteWord( t+32, u+4 );

	if( n == 128 ){
		for( i=0; i<26; i+=2 ){
			RotBlock( u+KIDX1[i+0], KSFT1[i+0], u+16 );
			RotBlock( u+KIDX1[i+1], KSFT1[i+1], u+18 );
			WordByte( u+16, e+i*8 );
		}
	}
	else{
		XorBlock( t+32, t+16, t+48 );

		Camellia_Feistel( t+48, SIGMA+32, t+56 );
		Camellia_Feistel( t+56, SIGMA+40, t+48 );

		ByteWord( t+16, u+8  );
		ByteWord( t+48, u+12 );

		for( i=0; i<34; i+=2 ){
			RotBlock( u+KIDX2[i+0], KSFT2[i+0], u+16 );
			RotBlock( u+KIDX2[i+1], KSFT2[i+1], u+18 );
			WordByte( u+16, e+(i<<3) );
		}
	}
}

void Camellia_Encrypt( const int n, const Byte *p, const Byte *e, Byte *c )
{//n=鍵長	p=平文	e=拡張鍵	c=暗号文
	int i;
```

```
	XorBlock( p, e+0, c );

	for( i=0; i<3; i++ ){
		Camellia_Feistel( c+0, e+16+(i<<4), c+8 );
		Camellia_Feistel( c+8, e+24+(i<<4), c+0 );
	}

	Camellia_FLlayer( c, e+64, e+72 );

	for( i=0; i<3; i++ ){
		Camellia_Feistel( c+0, e+80+(i<<4), c+8 );
		Camellia_Feistel( c+8, e+88+(i<<4), c+0 );
	}

	Camellia_FLlayer( c, e+128, e+136 );

	for( i=0; i<3; i++ ){
		Camellia_Feistel( c+0, e+144+(i<<4), c+8 );
		Camellia_Feistel( c+8, e+152+(i<<4), c+0 );
	}

	if( n == 128 ){
		SwapHalf( c );
		XorBlock( c, e+192, c );
	}
	else{
		Camellia_FLlayer( c, e+192, e+200 );

		for( i=0; i<3; i++ ){
			Camellia_Feistel( c+0, e+208+(i<<4), c+8 );
			Camellia_Feistel( c+8, e+216+(i<<4), c+0 );
		}

		SwapHalf( c );
		XorBlock( c, e+256, c );
	}
}

void Camellia_Decrypt( const int n, const Byte *c, const Byte *e, Byte *p )
{
	int i;

	if( n == 128 ){
		XorBlock( c, e+192, p );
	}
	else{
		XorBlock( c, e+256, p );

		for( i=2; i>=0; i-- ){
			Camellia_Feistel( p+0, e+216+(i<<4), p+8 );
			Camellia_Feistel( p+8, e+208+(i<<4), p+0 );
		}

		Camellia_FLlayer( p, e+200, e+192 );
	}

	for( i=2; i>=0; i-- ){
		Camellia_Feistel( p+0, e+152+(i<<4), p+8 );
```

```
		Camellia_Feistel( p+8, e+144+(i<<4), p+0 );
	}

	Camellia_FLlayer( p, e+136, e+128 );

	for( i=2; i>=0; i-- ){
		Camellia_Feistel( p+0, e+88+(i<<4), p+8 );
		Camellia_Feistel( p+8, e+80+(i<<4), p+0 );
	}

	Camellia_FLlayer( p, e+72, e+64 );

	for( i=2; i>=0; i-- ){
		Camellia_Feistel( p+0, e+24+(i<<4), p+8 );
		Camellia_Feistel( p+8, e+16+(i<<4), p+0 );
	}

	SwapHalf( p );
	XorBlock( p, e+0, p );
}

void Camellia_Feistel( const Byte *x, const Byte *k, Byte *y )
{
	Byte t[8];

	t[0] = SBOX1(x[0]^k[0]);
	t[1] = SBOX2(x[1]^k[1]);
	t[2] = SBOX3(x[2]^k[2]);
	t[3] = SBOX4(x[3]^k[3]);
	t[4] = SBOX2(x[4]^k[4]);
	t[5] = SBOX3(x[5]^k[5]);
	t[6] = SBOX4(x[6]^k[6]);
	t[7] = SBOX1(x[7]^k[7]);

	y[0] ^= t[0]^t[2]^t[3]^t[5]^t[6]^t[7];
	y[1] ^= t[0]^t[1]^t[3]^t[4]^t[6]^t[7];
	y[2] ^= t[0]^t[1]^t[2]^t[4]^t[5]^t[7];
	y[3] ^= t[1]^t[2]^t[3]^t[4]^t[5]^t[6];
	y[4] ^= t[0]^t[1]^t[5]^t[6]^t[7];
	y[5] ^= t[1]^t[2]^t[4]^t[6]^t[7];
	y[6] ^= t[2]^t[3]^t[4]^t[5]^t[7];
	y[7] ^= t[0]^t[3]^t[4]^t[5]^t[6];
}

void Camellia_FLlayer( Byte *x, const Byte *kl, const Byte *kr )
{
	Word t[4],u[4],v[4];

	ByteWord( x, t );
	ByteWord( kl, u );
	ByteWord( kr, v );

	t[1] ^= (t[0]&u[0])<<1^(t[0]&u[0])>>31;
	t[0] ^= t[1]|u[1];
	t[2] ^= t[3]|v[1];
	t[3] ^= (t[2]&v[0])<<1^(t[2]&v[0])>>31;

	WordByte( t, x );
}
```

```
void ByteWord( const Byte *x, Word *y )
{
	int i;
	for( i=0; i<4; i++ ){
		y[i] = ((Word)x[(i<<2)+0]<<24) + ((Word)x[(i<<2)+1]<<16)
			+ ((Word)x[(i<<2)+2]<<8 ) + ((Word)x[(i<<2)+3]<<0 );
	}
}

void WordByte( const Word *x, Byte *y )
{
	int i;
	for( i=0; i<4; i++ ){
		y[(i<<2)+0] = (Byte)(x[i]>>24&0xff);
		y[(i<<2)+1] = (Byte)(x[i]>>16&0xff);
		y[(i<<2)+2] = (Byte)(x[i]>> 8&0xff);
		y[(i<<2)+3] = (Byte)(x[i]>> 0&0xff);
	}
}

void RotBlock( const Word *x, const int n, Word *y )
{
	int r;
	if( r = (n & 31) ){
		y[0] = x[((n>>5)+0)&3]<<r^x[((n>>5)+1)&3]>>(32-r);
		y[1] = x[((n>>5)+1)&3]<<r^x[((n>>5)+2)&3]>>(32-r);
	}
	else{
		y[0] = x[((n>>5)+0)&3];
		y[1] = x[((n>>5)+1)&3];
	}
}

void SwapHalf( Byte *x )
{
	Byte t;
	int  i;
	for( i=0; i<8; i++ ){
		t = x[i];
		x[i] = x[8+i];
		x[8+i] = t;
	}
}

void XorBlock( const Byte *x, const Byte *y, Byte *z )
{
	int i;
	for( i=0; i<16; i++ ) z[i] = x[i] ^ y[i];
}


//////////////////////////////////////////////////////

// ECcml.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"
```

```
#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include "ECcml.h"
#include <stdio.h>

FILE *stream;
FILE *stream1;
FILE *stream2;

extern ushort EXTKEY[4][8];

// 暗号文の HEX 表示用
char toChar( int c )
{
        if( c >= 0 && c <= 9 )                  // 0～9 ならば
                return char( c + 0x30 );    // ASCII に変換して返す
        else if( c >= 10 && c <= 15 )       // 10～15 ならば
                return char( c + 0x37 );    // A～F の ASCII を返す
        else
                return ' ';
}

// メイン
int main(                                                       // 正常終了時:0、異常時:1
                int argc,                                       // 引数の数
                char** argv )                                   // 引数へのポインタ
{
        int i, len;
        int block;
        unsigned char klen[8], key[32], key2[64];
        unsigned char exkey[256];
        int numclosed;
        char j,k;
        char* bufp;
        char* bufc;
        int cnt;
        long mesLength; // 平文長（バイト）
        int c;

        printf( "Start!¥n" );

          // 引数チェック
        if( argc != 4 ){                                // 使い方の誤り
                printf( "引数の数が不正ですから異常終了します。¥n" );
                exit(1);
        }

    /* 鍵ファイルを開く */
    if( (stream  = fopen( argv[1], "rb" )) == NULL )
        printf( "Can not open Key file.¥n" );

   /* 平文を読み出すファイルを開く*/
    if( (stream1  = fopen( argv[2], "rb" )) == NULL )
        printf( "Can not open Plane Text file.¥n");

    /* 暗号文を書き込むファイルを開く */
    if( (stream2 = fopen( argv[3], "wb" )) == NULL )
```

```
        printf( "Can not open for Encrypted file.¥n" );


            fscanf(stream,"%s",klen);
            fscanf(stream,"%s",key2);
            len = atoi((char*)klen);

            for(i=0; i<len/8; i++){
                        j = *(key2+2*i);
                        k = *(key2+2*i+1);
                        if(j>=0x30 && j<=0x39) j = j-0x30;
                        else{
                                    if(j>=0x41 && j<=0x46) j = j-0x41+0x0A;
                        }
                        if(k>=0x30 && k<=0x39) k = k-0x30;
                        else{
                                    if(k>=0x41 && k<=0x46) k = k-0x41+0x0A;
                        }
                        key[i] = j*0x10 + k;
            }
            key[len/8] = NULL;

            Camellia_Ekeygen( len, key, exkey );

    // 平文
            fseek(stream1, 0, SEEK_END);
            long filelen = ftell(stream1);
            fseek(stream1,0,0);
            int head = sizeof(long);
            mesLength = filelen + head;

            //暗号化
            if(mesLength <= 1024){
                        block = mesLength/16 + ((mesLength%16)?1:0);
                        bufp = (char*)new(char[block*16 + 1]);
                        bufc = (char*)new(char[block*16 + 1]);
                        if(bufp == NULL){
                                    printf("メモリ不足です。異常終了します。¥r¥n");
                                    return(-1);
                                    }
                        (*(long*)(bufp)) = filelen;

            // 平文
                        fseek(stream1, 0, 0);
                        i = head;
                        do{
                                    c = fgetc(stream1);
                                    bufp[i]=c;
                                    i=i+1;
                        }while(c!=EOF);
                        bufp[i-1]=NULL;

        // 暗号化実行
            for(i=0;i<block;i++){
                Camellia_Encrypt( len,  (unsigned char*)(bufp+i*16), (unsigned char*)exkey, (unsigned
char*)(bufc+i*16));
                        //n=鍵長      bufp=平文      exkey=拡張鍵      ct=暗号文
            }
```

```
	// 暗号文を書き込む
		for( cnt = 0; cnt<block*16; cnt++ ){
			fwrite( &(bufc[cnt]),sizeof(char),1, stream2);
		}
	}
	else{
		block = mesLength/16 + ((mesLength%16)?1:0);
		bufp = (char*)new(char[1024 + 2]);
		bufc = (char*)new(char[1024 + 2]);
		if(bufp == NULL){
			printf("メモリ不足¥r¥n");
			return(-1);
			}
		(*(long*)(bufp)) = filelen;

		long rBlen = block;
		int r = 0;
		do{
		// 平文

			if(r == 0){
				i = head;
				fseek(stream1, r*1024, 0);
			}
			if(r > 0 ){
				i = 0;
				fseek(stream1, r*1024-4, 0);
			}
			do{
				c = fgetc(stream1);
				bufp[i]=c;
				i=i+1;
			}while((c!=EOF) && (i<=1024));
			bufp[i-1]=NULL;

			if(rBlen >= 1024/16){ block = 1024/16; }
			if(rBlen < 1024/16){ block = rBlen;}

		// 暗号化実行
	for(i=0;i<block;i++){
	    Camellia_Encrypt(len,    (unsigned char*)(bufp+i*16), (unsigned char*)exkey, (unsigned
char*)(bufc+i*16));
		//n=鍵長      bufp=平文      exkey=拡張鍵      ct=暗号文
	}

	// 暗号文を書き込む
			for( cnt = 0; cnt<block*16; cnt++ ){
				fwrite( &(bufc[cnt]),sizeof(char),1, stream2);
			}
			r += 1;
			rBlen -= 1024/16;
		}while(rBlen>0);
	}

	if( fclose( stream1 ) )
	printf( "Can not close Plane Text file.¥n" );
	/* ストリームを閉じる */
	if( fclose( stream2 ) )
		printf( "Can not close Encrypted file.¥n" );
```

```
    /* 他のすべてのファイルを閉じる */
        numclosed = _fcloseall( );

        delete[] bufp;
        delete[] bufc;

        printf( "End!¥n" );

        return 0;
}


//////////////////////////////////////////

// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//                  ECMisty.pch 生成されるプリコンパイル済ヘッダー
//                  stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。


//////////////////////////////////////////

/***
 * ECcml.h
 * Misty 暗号関数
 */

// 戻り値
#define  NOERROR                    0                   // エラーなし
#define  NOTENOUGHMEMORY -1                  // メモリー不足
#define  ACCESSERROR        -2               // アクセスエラー
#define  MATHERROR          -3               // 数学的な誤り
#define  KEYLENGTHERROR     -4               // 鍵長不正
#define OTHERERROR          -5               // その他のエラー

// 定数
#define MINKEYLENGTH        32               // 最低鍵長（ビット）


typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned char Byte;
typedef unsigned long Word;

/*********************/
/*** RSA 公開鍵暗号 ***/
/*********************/

// 暗号化    復号化
void Camellia_Ekeygen( const int, const Byte *, Byte * );
void Camellia_Encrypt( const int, const Byte *, const Byte *, Byte * );


//////////////////////////////////////
```

```
// stdafx.h : 標準のシステム インクルード ファイル、
//            または参照回数が多く、かつあまり変更されない
//            プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__0E1470F2_0311_4B61_AEFF_03F15B4BA312__INCLUDED_)
#define AFX_STDAFX_H__0E1470F2_0311_4B61_AEFF_03F15B4BA312__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_STDAFX_H__0E1470F2_0311_4B61_AEFF_03F15B4BA312__INCLUDED_)
```

１０．１．２　復号化部分

```
/*************************************************
 *                                               *
 *        Camellia Block Encryption Algorithm    *
 *          in ANSI-C Language : Camellia.c      *
 *                                               *
 *          Version M1.02 September 24 2001      *
 *   Copyright Mitsubishi Electric Corp 2000-2001   *
 *                                               *
 *************************************************/
#include "stdafx.h"

typedef unsigned char Byte;
typedef unsigned long Word;

void Camellia_Ekeygen( const int, const Byte *, Byte * );

void Camellia_Encrypt( const int, const Byte *, const Byte *, Byte * );
void Camellia_Decrypt( const int, const Byte *, const Byte *, Byte * );

void Camellia_Feistel( const Byte *, const Byte *, Byte * );
void Camellia_FLlayer( Byte *, const Byte *, const Byte * );

void ByteWord( const Byte *, Word * );
void WordByte( const Word *, Byte * );
void XorBlock( const Byte *, const Byte *, Byte * );
void SwapHalf( Byte * );
void RotBlock( const Word *, const int, Word * );

const Byte SIGMA[48] = {
0xa0,0x9e,0x66,0x7f,0x3b,0xcc,0x90,0x8b,
```

```
0xb6,0x7a,0xe8,0x58,0x4c,0xaa,0x73,0xb2,
0xc6,0xef,0x37,0x2f,0xe9,0x4f,0x82,0xbe,
0x54,0xff,0x53,0xa5,0xf1,0xd3,0x6f,0x1c,
0x10,0xe5,0x27,0xfa,0xde,0x68,0x2d,0x1d,
0xb0,0x56,0x88,0xc2,0xb3,0xe6,0xc1,0xfd};

const int KSFT1[26] = {
0,64,0,64,15,79,15,79,30,94,45,109,45,124,60,124,77,13,
94,30,94,30,111,47,111,47 };
const int KIDX1[26] = {
0,0,4,4,0,0,4,4,4,4,0,0,4,0,4,4,4,0,0,0,0,4,4,0,0,4,4 };
const int KSFT2[34] = {
0,64,0,64,15,79,15,79,30,94,30,94,45,109,45,109,60,124,
60,124,60,124,77,13,77,13,94,30,94,30,111,47,111,47 };
const int KIDX2[34] = {
0,0,12,12,8,8,4,4,8,8,12,12,0,0,4,4,0,0,8,8,12,12,
0,0,4,4,8,8,4,4,0,0,12,12 };

const Byte SBOX[256] = {
112,130, 44,236,179, 39,192,229,228,133, 87, 53,234, 12,174, 65,
 35,239,107,147, 69, 25,165, 33,237, 14, 79, 78, 29,101,146,189,
134,184,175,143,124,235, 31,206, 62, 48,220, 95, 94,197, 11, 26,
166,225, 57,202,213, 71, 93, 61,217,  1, 90,214, 81, 86,108, 77,
139, 13,154,102,251,204,176, 45,116, 18, 43, 32,240,177,132,153,
223, 76,203,194, 52,126,118,  5,109,183,169, 49,209, 23,  4,215,
 20, 88, 58, 97,222, 27, 17, 28, 50, 15,156, 22, 83, 24,242, 34,
254, 68,207,178,195,181,122,145, 36,  8,232,168, 96,252,105, 80,
170,208,160,125,161,137, 98,151, 84, 91, 30,149,224,255,100,210,
 16,196,  0, 72,163,247,117,219,138,  3,230,218,  9, 63,221,148,
135, 92,131,  2,205, 74,144, 51,115,103,246,243,157,127,191,226,
 82,155,216, 38,200, 55,198, 59,129,150,111, 75, 19,190, 99, 46,
233,121,167,140,159,110,188,142, 41,245,249,182, 47,253,180, 89,
120,152,  6,106,231, 70,113,186,212, 37,171, 66,136,162,141,250,
114,  7,185, 85,248,238,172, 10, 54, 73, 42,104, 60, 56,241,164,
 64, 40,211,123,187,201, 67,193, 21,227,173,244,119,199,128,158};

#define SBOX1(n) SBOX[(n)]
#define SBOX2(n) (Byte)((SBOX[(n)]>>7^SBOX[(n)]<<1)&0xff)
#define SBOX3(n) (Byte)((SBOX[(n)]>>1^SBOX[(n)]<<7)&0xff)
#define SBOX4(n) SBOX[((n)<<1^(n)>>7)&0xff]

void Camellia_Ekeygen( const int n, const Byte *k, Byte *e )
{
        Byte t[64];
        Word u[20];
        int  i;

        if( n == 128 ){
                for( i=0 ; i<16; i++ ) t[i] = k[i];
                for( i=16; i<32; i++ ) t[i] = 0;
        }
        else if( n == 192 ){
                for( i=0 ; i<24; i++ ) t[i] = k[i];
                for( i=24; i<32; i++ ) t[i] = k[i-8]^0xff;
        }
        else if( n == 256 ){
                for( i=0 ; i<32; i++ ) t[i] = k[i];
        }
```

```
	XorBlock( t+0, t+16, t+32 );

	Camellia_Feistel( t+32, SIGMA+0, t+40 );
	Camellia_Feistel( t+40, SIGMA+8, t+32 );

	XorBlock( t+32, t+0, t+32 );

	Camellia_Feistel( t+32, SIGMA+16, t+40 );
	Camellia_Feistel( t+40, SIGMA+24, t+32 );

	ByteWord( t+0,   u+0 );
	ByteWord( t+32, u+4 );

	if( n == 128 ){
		for( i=0; i<26; i+=2 ){
			RotBlock( u+KIDX1[i+0], KSFT1[i+0], u+16 );
			RotBlock( u+KIDX1[i+1], KSFT1[i+1], u+18 );
			WordByte( u+16, e+i*8 );
		}
	}
	else{
		XorBlock( t+32, t+16, t+48 );

		Camellia_Feistel( t+48, SIGMA+32, t+56 );
		Camellia_Feistel( t+56, SIGMA+40, t+48 );

		ByteWord( t+16, u+8   );
		ByteWord( t+48, u+12 );

		for( i=0; i<34; i+=2 ){
			RotBlock( u+KIDX2[i+0], KSFT2[i+0], u+16 );
			RotBlock( u+KIDX2[i+1], KSFT2[i+1], u+18 );
			WordByte( u+16, e+(i<<3) );
		}
	}
}

void Camellia_Encrypt( const int n, const Byte *p, const Byte *e, Byte *c )
{//n=鍵長      p=平文      e=拡張鍵      c=暗号文
	int i;

	XorBlock( p, e+0, c );

	for( i=0; i<3; i++ ){
		Camellia_Feistel( c+0, e+16+(i<<4), c+8 );
		Camellia_Feistel( c+8, e+24+(i<<4), c+0 );
	}

	Camellia_FLlayer( c, e+64, e+72 );

	for( i=0; i<3; i++ ){
		Camellia_Feistel( c+0, e+80+(i<<4), c+8 );
		Camellia_Feistel( c+8, e+88+(i<<4), c+0 );
	}

	Camellia_FLlayer( c, e+128, e+136 );

	for( i=0; i<3; i++ ){
		Camellia_Feistel( c+0, e+144+(i<<4), c+8 );
```

```
		Camellia_Feistel( c+8, e+152+(i<<4), c+0 );
	}

	if( n == 128 ){
		SwapHalf( c );
		XorBlock( c, e+192, c );
	}
	else{
		Camellia_FLlayer( c, e+192, e+200 );

		for( i=0; i<3; i++ ){
			Camellia_Feistel( c+0, e+208+(i<<4), c+8 );
			Camellia_Feistel( c+8, e+216+(i<<4), c+0 );
		}

		SwapHalf( c );
		XorBlock( c, e+256, c );
	}
}

void Camellia_Decrypt( const int n, const Byte *c, const Byte *e, Byte *p )
{
	int i;

	if( n == 128 ){
		XorBlock( c, e+192, p );
	}
	else{
		XorBlock( c, e+256, p );

		for( i=2; i>=0; i-- ){
			Camellia_Feistel( p+0, e+216+(i<<4), p+8 );
			Camellia_Feistel( p+8, e+208+(i<<4), p+0 );
		}

		Camellia_FLlayer( p, e+200, e+192 );
	}

	for( i=2; i>=0; i-- ){
		Camellia_Feistel( p+0, e+152+(i<<4), p+8 );
		Camellia_Feistel( p+8, e+144+(i<<4), p+0 );
	}

	Camellia_FLlayer( p, e+136, e+128 );

	for( i=2; i>=0; i-- ){
		Camellia_Feistel( p+0, e+88+(i<<4), p+8 );
		Camellia_Feistel( p+8, e+80+(i<<4), p+0 );
	}

	Camellia_FLlayer( p, e+72, e+64 );

	for( i=2; i>=0; i-- ){
		Camellia_Feistel( p+0, e+24+(i<<4), p+8 );
		Camellia_Feistel( p+8, e+16+(i<<4), p+0 );
	}

	SwapHalf( p );
	XorBlock( p, e+0, p );
```

```
}

void Camellia_Feistel( const Byte *x, const Byte *k, Byte *y )
{
	Byte t[8];

	t[0] = SBOX1(x[0]^k[0]);
	t[1] = SBOX2(x[1]^k[1]);
	t[2] = SBOX3(x[2]^k[2]);
	t[3] = SBOX4(x[3]^k[3]);
	t[4] = SBOX2(x[4]^k[4]);
	t[5] = SBOX3(x[5]^k[5]);
	t[6] = SBOX4(x[6]^k[6]);
	t[7] = SBOX1(x[7]^k[7]);

	y[0] ^= t[0]^t[2]^t[3]^t[5]^t[6]^t[7];
	y[1] ^= t[0]^t[1]^t[3]^t[4]^t[6]^t[7];
	y[2] ^= t[0]^t[1]^t[2]^t[4]^t[5]^t[7];
	y[3] ^= t[1]^t[2]^t[3]^t[4]^t[5]^t[6];
	y[4] ^= t[0]^t[1]^t[5]^t[6]^t[7];
	y[5] ^= t[1]^t[2]^t[4]^t[6]^t[7];
	y[6] ^= t[2]^t[3]^t[4]^t[5]^t[7];
	y[7] ^= t[0]^t[3]^t[4]^t[5]^t[6];
}

void Camellia_FLlayer( Byte *x, const Byte *kl, const Byte *kr )
{
	Word t[4],u[4],v[4];

	ByteWord( x, t );
	ByteWord( kl, u );
	ByteWord( kr, v );

	t[1] ^= (t[0]&u[0])<<1^(t[0]&u[0])>>31;
	t[0] ^= t[1]|u[1];
	t[2] ^= t[3]|v[1];
	t[3] ^= (t[2]&v[0])<<1^(t[2]&v[0])>>31;

	WordByte( t, x );
}

void ByteWord( const Byte *x, Word *y )
{
	int i;
	for( i=0; i<4; i++ ){
		y[i] = ((Word)x[(i<<2)+0]<<24) + ((Word)x[(i<<2)+1]<<16)
			+ ((Word)x[(i<<2)+2]<<8 ) + ((Word)x[(i<<2)+3]<<0 );
	}
}

void WordByte( const Word *x, Byte *y )
{
	int i;
	for( i=0; i<4; i++ ){
		y[(i<<2)+0] = (Byte)(x[i]>>24&0xff);
		y[(i<<2)+1] = (Byte)(x[i]>>16&0xff);
		y[(i<<2)+2] = (Byte)(x[i]>> 8&0xff);
		y[(i<<2)+3] = (Byte)(x[i]>> 0&0xff);
	}
```

```
}

void RotBlock( const Word *x, const int n, Word *y )
{
	int r;
	if( r = (n & 31) ){
		y[0] = x[((n>>5)+0)&3]<<r^x[((n>>5)+1)&3]>>(32-r);
		y[1] = x[((n>>5)+1)&3]<<r^x[((n>>5)+2)&3]>>(32-r);
	}
	else{
		y[0] = x[((n>>5)+0)&3];
		y[1] = x[((n>>5)+1)&3];
	}
}

void SwapHalf( Byte *x )
{
	Byte t;
	int  i;
	for( i=0; i<8; i++ ){
		t = x[i];
		x[i] = x[8+i];
		x[8+i] = t;
	}
}

void XorBlock( const Byte *x, const Byte *y, Byte *z )
{
	int i;
	for( i=0; i<16; i++ ) z[i] = x[i] ^ y[i];
}


////////////////////////////////////////////////

// DCcml.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include "DCcml.h"
#include <stdio.h>

FILE *stream;
FILE *stream1;
FILE *stream2;

extern ushort EXTKEY[4][8];

// 暗号文の HEX 表示用
char toChar( int c )
{
	if( c >= 0 && c <= 9 )				// 0～9 ならば
		return char( c + 0x30 );	// ASCII に変換して返す
	else if( c >= 10 && c <= 15 )		// 10～15 ならば
		return char( c + 0x37 );	// A～F の ASCII を返す
```

```
	else
		return ' ';
}

// メイン
int main(                                                    // 正常終了時:0、異常時:1
		int argc,                                    // 引数の数
		char** argv )                                // 引数へのポインタ
{
	int i, len;
	int block;
	char j,k;
	unsigned char klen[8], key[32], key2[64];
	unsigned char exkey[256];
	int numclosed;
	int cnt;
	int lenp;
	int c;
	char* bufp;
	unsigned char* bufc;

	printf( "Start!¥n" );

	// 引数チェック
	if( argc != 4 ){                             // 使い方の誤り
		exit(1);
	}

   /* 鍵ファイルを開く */
   if( (stream  = fopen( argv[1], "rb" )) == NULL )
       printf( "Can not open Key file.¥n" );

  /* 平文を書き込むファイルを開く*/
   if( (stream1  = fopen( argv[3], "wb" )) == NULL )
       printf( "Can not open file for Plane Text.¥n");

   /* 暗号文を読み出すファイルを開く */
   if( (stream2 = fopen( argv[2], "rb" )) == NULL )
       printf( "Can not open Encrypted file.¥n" );


	fseek(stream,0,0);
	fscanf(stream,"%s",klen);
	fscanf(stream,"%s",key2);
	len = atoi((char*)klen);

	for(i=0; i<len/8; i++){
		j = *(key2+2*i);
		k = *(key2+2*i+1);
		if(j>=0x30 && j<=0x39) j = j-0x30;
		else{
			if(j>=0x41 && j<=0x46) j = j-0x41+0x0A;
		}
		if(k>=0x30 && k<=0x39) k = k-0x30;
		else{
			if(k>=0x41 && k<=0x46) k = k-0x41+0x0A;
		}
		key[i] = j*0x10 + k;
	}
```

```
        key[len/8] = NULL;

        Camellia_Ekeygen( len, key, exkey );

    // 暗号文
        fseek(stream2, 0, SEEK_END);
        long filelen = ftell(stream2);
        fseek(stream2,0,0);
        int head = sizeof(long);
        long mesLength = filelen;

        if(mesLength <= 1024){
                block = mesLength/16 + ((mesLength%16)?1:0);
                bufc = (unsigned char*)new(char[block*16 + 2]);
                bufp = (char*)new(char[block*16 + 2]);
                if(bufp == NULL){
                        printf("メモリ不足¥r¥n");
                        return(-1);
                        }

        // 暗文
                fseek(stream2, 0, 0);
                i = 0;
                do{
                        c = fgetc(stream2);
                        bufc[i]=c;
                        i=i+1;
                }while(c!=EOF);
                bufc[i-1]=NULL;

     // 復号化実行
                for(i=0;i<block;i++){
                        Camellia_Decrypt( len,      (unsigned   char*)(bufc+i*16),   exkey,   (unsigned
char*)(bufp+i*16));
                    //n=鍵長       bufc=暗号文       exkey=拡張鍵       bufp=平文
                }

        // 復号文を書き込む
                lenp = *((long*)(bufp));
                bufp[lenp+head] = NULL;
                for( cnt = 0; cnt<lenp; cnt++ ){
                        fwrite( &(bufp[cnt+head]),sizeof(char),1, stream1);
                }
        }
        else{
                block = mesLength/16 + ((mesLength%16)?1:0);
                bufc = (unsigned char*)new(char[1024 + 2]);
                bufp = (char*)new(char[1024 + 2]);
                if(bufp == NULL){
                        printf("メモリ不足¥r¥n");
                        return(-1);
                        }
                long rBlen = block;
                int r = 0;
                do{
                        // 暗文
                        fseek(stream2, r*1024, 0);
                        i = 0;
                        do{
```

```
                                        c = fgetc(stream2);
                                        bufc[i]=c;
                                        i=i+1;
                                }while((c!=EOF) && (i<=1024));
                                bufc[i-1]=NULL;

                                if(rBlen >= 1024/16){ block = 1024/16; }
                                if(rBlen < 1024/16){ block = rBlen;}

                        // 復号化実行
                                for(i=0;i<block;i++){
                                        Camellia_Decrypt( len, (unsigned char*)(bufc+i*16), exkey, (unsigned
char*)(bufp+i*16) );
                                }

                        // 復号文を書き込む
                                if(r==0){
                                        lenp = (*(long*)(bufp));
                                        for( cnt = 0; cnt<(1024-head); cnt++ ){
                                                fwrite( &(bufp[cnt+head]),sizeof(char),1, stream1);
                                        }
                                        lenp -= 1024-head;
                                }
                                else{
                                        if(rBlen >= 1024/16){
                                                for( cnt = 0; cnt<1024; cnt++ ){
                                                        fwrite( &(bufp[cnt]),sizeof(char),1, stream1);
                                                }
                                                lenp -= 1024;
                                        }
                                        else{
                                                for( cnt = 0; cnt<lenp; cnt++ ){
                                                        fwrite( &(bufp[cnt]),sizeof(char),1, stream1);
                                                }
                                        }
                                }
                                r += 1;
                                rBlen -= 1024/16;
                        }while(rBlen>0);
                }

                if( fclose( stream1 ) )
                printf("平文用ファイルは閉じられませんでした。¥n");
                if( fclose( stream2 ) )
                        printf("暗号文のファイルは閉じられませんでした。¥n");
        /* 他のすべてのファイルを閉じる */
                numclosed = _fcloseall( );

                delete[] bufp;
                delete[] bufc;

                printf( "End!¥n" );

                return 0;
}


//////////////////////////////////////////////////////
```

```
// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//                  DCMisty.pch 生成されるプリコンパイル済ヘッダー
//                  stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。

////////////////////////////////////////

/***
 * DCcml.h
 * Camellia 暗号関数
 */

// 戻り値
#define  NOERROR                0                    // エラーなし
#define  NOTENOUGHMEMORY -1                          // メモリー不足
#define  ACCESSERROR        -2                       // アクセスエラー
#define  MATHERROR          -3                       // 数学的な誤り
#define  KEYLENGTHERROR     -4                       // 鍵長不正
#define OTHERERROR          -5                       // その他のエラー

// 定数
#define MINKEYLENGTH        32                       // 最低鍵長（ビット）


typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned char Byte;
typedef unsigned long Word;

/**********************/
/*** RSA 公開鍵暗号 ***/
/**********************/

// 暗号化    復号化
void Camellia_Ekeygen( const int, const Byte *, Byte * );
void Camellia_Decrypt( const int, const Byte *, const Byte *, Byte * );


////////////////////////////////////////

// stdafx.h : 標準のシステム インクルード ファイル、
//                  または参照回数が多く、かつあまり変更されない
//                  プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__0970D4A0_4770_4806_B6B8_C06A5B282D03__INCLUDED_)
#define AFX_STDAFX_H__0970D4A0_4770_4806_B6B8_C06A5B282D03__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します
```

```
#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_STDAFX_H__0970D4A0_4770_4806_B6B8_C06A5B282D03__INCLUDED_)
```

１０．１．３　鍵の形式

```
128
971B85D2FF30A1B20371353210EA2C61
```

１０．２　AES 暗号ソースコード

参考文献：The Design of Rijndael，　AES - The Advanced Encryption Standard，Springer のソースコードには 2 箇所だけ誤りがあります。これに関しては著者が訂正しています。このソースコードを変形して、Protected Communication System　で利用する形にしたものが以下のソースコードです。

この変形が安全性に影響を与えないと考えていますが、皆様自身で参考文献のソースコードとここに記載されているソースコードを比較されて安全性を確認してください。

１０．２．１　暗号化部分

```
// RijnEncrypt.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include <stdio.h>
#include <time.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <malloc.h>

#define file_len(x) (unsigned long)x
#define MAXBC 8
#define MAXKC 8
#define MAXROUNDS 14

typedef unsigned char word8;
typedef unsigned int word32;

int BC, KC, ROUNDS;
int s;

word8 Logtable[256] = {
            0,   0, 25,   1, 50,   2, 26,198, 75,199, 27,104, 51,238,223,   3,
          100,   4,224, 14, 52,141,129,239, 76,113,   8,200,248,105, 28,193,
          125,194, 29,181,249,185, 39,106, 77,228,166,114,154,201,   9,120,
          101, 47,138,   5, 33, 15,225, 36, 18,240,130, 69, 53,147,218,142,
          150,143,219,189, 54,208,206,148, 19, 92,210,241, 64, 70,131, 56,
          102,221,253, 48,191,   6,139, 98,179, 37,226,152, 34,136,145, 16,
          126,110, 72,195,163,182, 30, 66, 58,107, 40, 84,250,133, 61,186,
           43,121, 10, 21,155,159, 94,202, 78,212,172,229,243,115,167, 87,
          175, 88,168, 80,244,234,214,116, 79,174,233,213,231,230,173,232,
           44,215,117,122,235, 22, 11,245, 89,203, 95,176,156,169, 81,160,
```

```
        127, 12,246,111, 23,196, 73,236,216, 67, 31, 45,164,118,123,183,
        204,187, 62, 90,251, 96,177,134, 59, 82,161,108,170, 85, 41,157,
        151,178,135,144, 97,190,220,252,188,149,207,205, 55, 63, 91,209,
         83, 57,132, 60, 65,162,109, 71, 20, 42,158, 93, 86,242,211,171,
         68, 17,146,217, 35, 32, 46,137,180,124,184, 38,119,153,227,165,
        103, 74,237,222,197, 49,254, 24, 13, 99,140,128,192,247,112,  7};

word8 Alogtable[256] = {
          1,  3,  5, 15, 17, 51, 85,255, 26, 46,114,150,161,248, 19, 53,
         95,225, 56, 72,216,115,149,164,247,  2,  6, 10, 30, 34,102,170,
        229, 52, 92,228, 55, 89,235, 38,106,190,217,112,144,171,230, 49,
         83,245,  4, 12, 20, 60, 68,204, 79,209,104,184,211,110,178,205,
         76,212,103,169,224, 59, 77,215, 98,166,241,  8, 24, 40,120,136,
        131,158,185,208,107,189,220,127,129,152,179,206, 73,219,118,154,
        181,196, 87,249, 16, 48, 80,240, 11, 29, 39,105,187,214, 97,163,
        254, 25, 43,125,135,146,173,236, 47,113,147,174,233, 32, 96,160,
        251, 22, 58, 78,210,109,183,194, 93,231, 50, 86,250, 21, 63, 65,
        195, 94,226, 61, 71,201, 64,192, 91,237, 44,116,156,191,218,117,
        159,186,213,100,172,239, 42,126,130,157,188,223,122,142,137,128,
        155,182,193, 88,232, 35,101,175,234, 37,111,177,200, 67,197, 84,
        252, 31, 33, 99,165,244,  7,  9, 27, 45,119,153,176,203, 70,202,
         69,207, 74,222,121,139,134,145,168,227, 62, 66,198, 81,243, 14,
         18, 54, 90,238, 41,123,141,140,143,138,133,148,167,242, 13, 23,
         57, 75,221,124,132,151,162,253, 28, 36,108,180,199, 82,246,  1};

word8 S[256] = {
         99,124,119,123,242,107,111,197, 48,  1,103, 43,254,215,171,118,
        202,130,201,125,250, 89, 71,240,173,212,162,175,156,164,114,192,
        183,253,147, 38, 54, 63,247,204, 52,165,229,241,113,216, 49, 21,
          4,199, 35,195, 24,150,  5,154,  7, 18,128,226,235, 39,178,117,
          9,131, 44, 26, 27,110, 90,160, 82, 59,214,179, 41,227, 47,132,
         83,209,  0,237, 32,252,177, 91,106,203,190, 57, 74, 76, 88,207,
        208,239,170,251, 67, 77, 51,133, 69,249,  2,127, 80, 60,159,168,
         81,163, 64,143,146,157, 56,245,188,182,218, 33, 16,255,243,210,
        205, 12, 19,236, 95,151, 68, 23,196,167,126, 61,100, 93, 25,115,
         96,129, 79,220, 34, 42,144,136, 70,238,184, 20,222, 94, 11,219,
        224, 50, 58, 10, 73,  6, 36, 92,194,211,172, 98,145,149,228,121,
        231,200, 55,109,141,213, 78,169,108, 86,244,234,101,122,174,  8,
        186,120, 37, 46, 28,166,180,198,232,221,116, 31, 75,189,139,138,
        112, 62,181,102, 72,  3,246, 14, 97, 53, 87,185,134,193, 29,158,
        225,248,152, 17,105,217,142,148,155, 30,135,233,206, 85, 40,223,
        140,161,137, 13,191,230, 66,104, 65,153, 45, 15,176, 84,187, 22};

word8 Si[256] = {
         82,  9,106,213, 48, 54,165, 56,191, 64,163,158,129,243,215,251,
        124,227, 57,130,155, 47,255,135, 52,142, 67, 68,196,222,233,203,
         84,123,148, 50,166,194, 35, 61,238, 76,149, 11, 66,250,195, 78,
          8, 46,161,102, 40,217, 36,178,118, 91,162, 73,109,139,209, 37,
        114,248,246,100,134,104,152, 22,212,164, 92,204, 93,101,182,146,
        108,112, 72, 80,253,237,185,218, 94, 21, 70, 87,167,141,157,132,
        144,216,171,  0,140,188,211, 10,247,228, 88,  5,184,179, 69,  6,
        208, 44, 30,143,202, 63, 15,  2,193,175,189,  3,  1, 19,138,107,
         58,145, 17, 65, 79,103,220,234,151,242,207,206,240,180,230,115,
        150,172,116, 34,231,173, 53,133,226,249, 55,232, 28,117,223,110,
         71,241, 26,113, 29, 41,197,137,111,183, 98, 14,170, 24,190, 27,
        252, 86, 62, 75,198,210,121, 32,154,219,192,254,120,205, 90,244,
         31,221,168, 51,136,  7,199, 49,177, 18, 16, 89, 39,128,236, 95,
         96, 81,127,169, 25,181, 74, 13, 45,229,122,159,147,201,156,239,
        160,224, 59, 77,174, 42,245,176,200,235,187, 60,131, 83,153, 97,
```

```
        23, 43,    4,126,186,119,214, 38,225,105, 20, 99, 85, 33, 12,125};


word32 RC[30] = {
        0x00,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,
        0x1b,0x36,0x6c,0xd8,0xab,0x4d,0x9a,0x2f,0x5e,
        0xbc,0x63,0xc6,0x97,0x35,0x6a,0xd4,0xb3,0x7d,
        0xfa,0xef,0xc5};



static word8 shifts[5][4] = {
        0,1,2,3,
        0,1,2,3,
        0,1,2,3,
        0,1,2,4,
        0,1,3,4};

static int numrounds[5][5] = {
        10,11,12,13,14,
        11,11,12,13,14,
        12,12,12,13,14,
        13,13,13,13,14,
        14,14,14,14,14};



word8 mul(word8 a, word8 b) {
        if(a && b) return Alogtable[(Logtable[a] + Logtable[b])%255];
        else return 0;
}

void AddRoundKey(word8 a[4][MAXBC], word8 rk[4][MAXBC]) {
        int i, j;

        for(i = 0; i<4; i++)
                for(j=0;j<BC;j++) a[i][j] ^= rk[i][j];
}

void SubBytes(word8 a[4][MAXBC], word8 box[256]){
        int i,j;

        for(i=0;i<4;i++)
                for(j=0;j<BC;j++) a[i][j] = box[a[i][j]] ;
}

void ShiftRows(word8 a[4][MAXBC], word8 d){
        word8 tmp[MAXBC];
        int i,j;

        if(d==0){
                for(i=1;i<4;i++){
                        for(j=0;j<BC;j++)
                                tmp[j] = a[i][(j+shifts[BC-4][i]) % BC];
                        for(j=0;j<BC;j++) a[i][j] = tmp[j];
                }
        }
        else{
                for(i=1;i<4;i++){
```

```
                        for(j=0;j<BC;j++)
                                tmp[j] = a[i][(BC+j-shifts[BC-4][i]) % BC];
                        for(j=0;j<BC;j++) a[i][j] = tmp[j];
                }
        }
}

void MixColumns(word8 a[4][MAXBC]){
        word8 b[4][MAXBC];
        int i,j;

        for(j=0;j<BC;j++)
                for(i=0;i<4;i++)
                        b[i][j] = mul(2,a[i][j])
                                ^ mul(3,a[(i+1) % 4][j])
                                ^ a[(i+2) % 4][j]
                                ^ a[(i+3) % 4][j];
        for(i=0;i<4;i++)
                for(j=0;j<BC;j++) a[i][j] = b[i][j];
}

void InvMixColumns(word8 a[4][MAXBC]){
        word8 b[4][MAXBC];
        int i,j;

        for(j=0;j<BC;j++)
        for(i=0;i<4;i++)
                b[i][j] = mul(0xe, a[i][j])
                        ^ mul(0xb, a[(i+1) % 4][j])
                        ^ mul(0xd, a[(i+2) % 4][j])
                        ^ mul(0x9, a[(i+3) % 4][j]);
        for(i=0;i<4;i++)
                for(j=0;j<BC;j++) a[i][j] = b[i][j];
}

int KeyExpansion(word8 k[4][MAXKC],
                                word8 W[MAXROUNDS+1][4][MAXBC]){
        int i,j,t,RCpointer = 1;
        word8 tk[4][MAXKC];

        for(j=0;j<KC;j++)
                for(i=0;i<4;i++)
                        tk[i][j] = k[i][j];
        t = 0;

        for(j=0;(j<KC) && (t<(ROUNDS+1)*BC); j++, t++)
                for(i=0;i<4;i++) W[t / BC][i][t % BC] = tk[i][j];

        while(t<(ROUNDS+1)*BC){
                for(i=0;i<4;i++)
                        tk[i][0] ^= S[tk[(i+1)%4][KC-1]];
                tk[0][0] ^= RC[RCpointer++];

                if(KC<=6)
                        for(j=1;j<KC;j++)
                                for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
                else{
                        for(j=1;j<4;j++)
                                for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
```

```
                        for(i=0;i<4;i++) tk[i][4] ^= S[tk[i][3]];
                        for(j=5;j<KC;j++)
                                for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
                }
                for(j=0; (j<KC) && (t<(ROUNDS+1)*BC); j++, t++)
                        for(i=0;i<4;i++) W[t/BC][i][t%BC] = tk[i][j];
        }
        return 0;
}

int Encrypt(word8 a[4][MAXBC], word8 rk[MAXROUNDS+1][4][MAXBC]){
        int r;

        AddRoundKey(a,rk[0]);

        for(r=1;r<ROUNDS;r++){
                SubBytes(a,S);
                ShiftRows(a,0);
                MixColumns(a);
                AddRoundKey(a,rk[r]);
        }

        SubBytes(a,S);
        ShiftRows(a,0);
        AddRoundKey(a,rk[ROUNDS]);

        return 0;
}
/*
int Decrypt(word8 a[4][MAXBC], word8 rk[MAXROUNDS+1][4][MAXBC]){
        int r;

        AddRoundKey(a,rk[ROUNDS]);
        SubBytes(a,Si);
        ShiftRows(a,1);

        for(r=ROUNDS-1;r>0;r--){
                AddRoundKey(a,rk[r]);
                InvMixColumns(a);
                SubBytes(a,Si);
                ShiftRows(a,1);
        }
        AddRoundKey(a,rk[0]);

        return 0;
}
*/

int main(int argc, char* argv[])
{
        int i,j,klen,blen;
        word8 a[4][MAXBC], rk[MAXROUNDS+1][4][MAXBC], sk[4][MAXKC];
        char c_klen[8], c_blen[8], pass[128];
        FILE *fkey =0, *fin = 0, *fout = 0;
    fpos_t flen;
        char *dbuf;
    unsigned long len, rlen, blen4;

        printf( "Start!¥n" );
```

```
	if( argc != 4 ){                                    // 使い方の誤り
		printf( "argc != 4 ¥n" );
		return -1;
	}

	if((fkey = fopen(argv[1], "rt")) == NULL){
		printf("Can not find key file for encryption. ¥n");
		return (-1);
	}

	fgets( c_klen, 6, fkey );
	fgets( c_blen, 6, fkey );
	fgets( pass, 127, fkey );

	klen = atoi(c_klen)/32;
	blen = atoi(c_blen)/32;
	blen4 = (unsigned int)blen*4;

	KC = klen;
	if(KC<4 || 8<KC){
		printf("Wrong key size. ¥n");
		return (-1);
	}

	BC = blen;
	if(BC<4 || 8<BC){
		printf("Wrong block size. ¥n");
		return (-1);
	}

	dbuf = (char *)malloc(2*MAXBC*4);
	if(dbuf == NULL){
		printf("No memory. ¥n");
		return (-1);
	}

	if((fin = fopen(argv[2], "rb")) == NULL){
		printf("Can not open plane text file. ¥n");
		return (-1);
	}
fseek(fin, 0, SEEK_END);
fgetpos(fin, &flen);
rlen = file_len(flen);
// reset to start
fseek(fin, 0, SEEK_SET);

	if((fout = fopen(argv[3], "wb")) == NULL){
		printf("Can not open encrypted data file. ¥n");
		return (-1);
	}

	ROUNDS = numrounds[KC-4][BC-4];

	int k = 0;
	for(j=0;j<KC;j++){
		for(i=0;i<4;i++){
			sk[i][j] = pass[k];
			k += 1;
```

```
                }
        }
        KeyExpansion(sk,rk);

////////////////////////////////////////////////////////
        s = sizeof(unsigned int);
        // write the bytes of the file
        *((unsigned int*)dbuf) = rlen;
        len = (unsigned long) fread(dbuf+s, 1, blen4-s , fin);
    rlen -= len;
    // pad the file bytes with zeroes
    for(i = len+s; (unsigned int)i < blen4 ; ++i)
        dbuf[i] = 0;
    // encrypt the top 16 bytes of the buffer
        k=0;
        for(j=0;j<BC;j++){
                for(i=0;i<4;i++){
                        a[i][j] = dbuf[k];
                        k++;
                }
        }
        Encrypt(a,rk);
        k=0;
        for(j=0;j<BC;j++){
                for(i=0;i<4;i++){
                        dbuf[k] = a[i][j];
                        k++;
                }
        }
    if(fwrite(dbuf, 1, blen4, fout) != (unsigned int)blen4)
        return -2;


    if((rlen <= (unsigned int)blen4) && (rlen > 0))
    {   // if the file length is less than or equal to 16 bytes
        // read the bytes of the file into the buffer and verify length
        len = (unsigned long) fread(dbuf, 1, blen4 , fin);
        rlen -= len;

        if(rlen > 0)
            return -1;

        // pad the file bytes with zeroes
        for(i = len; (unsigned int)i < blen4 ; ++i)
            dbuf[i] = 0;

        // encrypt the top 16 bytes of the buffer
                k=0;
                for(j=0;j<BC;j++){
                        for(i=0;i<4;i++){
                                a[i][j] = dbuf[k];
                                k++;
                        }
                }

                Encrypt(a,rk);

        // write the IV and the encrypted file bytes
                k=0;
```

```
                for(j=0;j<BC;j++){
                        for(i=0;i<4;i++){
                                dbuf[k] = a[i][j];
                                k++;
                        }
                }
        if(fwrite(dbuf, 1, blen4, fout) != (unsigned int)blen4)
            return -2;
}
else
{    // if the file length is more 16 bytes
     // read the file a block at a time
     while(rlen > 0 && !feof(fin))
     {
          // read a block and reduce the remaining byte count
          len = (unsigned long)fread(dbuf, 1, blen4 , fin);
          rlen -= len;

          // verify length of block
          if(len != (unsigned int)blen4 ){
          // pad the file bytes with zeroes
                                for(i = len; (unsigned int)i < blen4 ; ++i)
                                        dbuf[i] = 0;
                        }

          // encrypt the block
                        k=0;
                        for(j=0;j<BC;j++){
                                for(i=0;i<4;i++){
                                        a[i][j] = dbuf[k];
                                        k++;
                                }
                        }
                        Encrypt(a,rk);
          // write the encrypted block
                        k=0;
                        for(j=0;j<BC;j++){
                                for(i=0;i<4;i++){
                                        dbuf[k] = a[i][j];
                                        k++;
                                }
                        }
          if(fwrite(dbuf, 1, blen4, fout) != blen4)
              return -2;

          // if there is only one more block do ciphertext stealing
          if(rlen > 0 && rlen < (unsigned int)blen4 )
          {
                        len = (unsigned long)fread(dbuf, 1, blen4 , fin);

                    // clear the remainder of the bottom half of buffer
          for(i = 0; (unsigned int)i < (unsigned int)blen4    - len; ++i)
                dbuf[len +i] = 0;

                // encrypt the final block
                        k=0;
                        for(j=0;j<BC;j++){
                                for(i=0;i<4;i++){
                                        a[i][j] = dbuf[k];
```

```
                                        k++;
                                }
                        }
                        Encrypt(a,rk);

                        k=0;
                        for(j=0;j<BC;j++){
                                for(i=0;i<4;i++){
                                        dbuf[k] = a[i][j];
                                        k++;
                                }
                        }
                if(fwrite(dbuf, 1, blen4, fout) != blen4)
                    return -2;

                        break;
                    // set the length of the final write
                }

        }
    }

        if(fkey)
        {
        fclose(fkey);
        }

        if(fout)
        {
        fclose(fout);
        }

        if(fin)
        {
                fclose(fin);
        }

        free(dbuf);

  ////////////////////////////////////////////////
        printf( "End!¥n" );

        return 0;
}


//////////////////////////////////////////////////////////////

// stdafx.cpp：標準インクルードを含むソース ファイルです。
// Crytest.pch は、プリコンパイル済みヘッダーになります。
// stdafx.obj にはプリコンパイル済み方情報が含まれます。

#include "stdafx.h"


//////////////////////////////////////////////////////

// stdafx.h：標準のシステム インクルード ファイルのインクルード ファイル、または
// 参照回数が多く、かつあまり変更されない、プロジェクト専用のインクルード ファイル
```

// を記述します。

```
#pragma once

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN            // Windows ヘッダーから使用されていない部分を除外します。
#endif

// 下で指定された定義の前に対象プラットフォームを指定しなければならない場合、以下の定義を変更してください。
// 異なるプラットフォームに対応する値に関する最新情報については、MSDN を参照してください。
#ifndef WINVER                          // Windows 95 および Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define WINVER 0x0400           // これを Windows 98 および Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINNT            // Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINNT 0x0400             // これを Windows 98 および Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINDOWS                  // Windows 98 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINDOWS 0x0500 // これを Windows Me またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_IE                       // IE 4.0 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_IE 0x0400        // これを IE 5.0  またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS      // 一部の CString コンストラクタは明示的です。

// 一般的で無視しても安全な MFC の警告メッセージの一部の非表示を解除します。
#define _AFX_ALL_WARNINGS

#include <afxwin.h>             // MFC のコアおよび標準コンポーネント
#include <afxext.h>             // MFC の拡張部分

#include <afxdtctl.h>           // MFC の Internet Explorer 4 コモン コントロール サポート
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC の Windows コモン コントロール サポート
#endif // _AFX_NO_AFXCMN_SUPPORT
```

１０．２．２　復号化部分

```
// RijnDecrypt.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"
```

```
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <malloc.h>

#define file_len(x) (unsigned long)x
#define MAXBC 8
#define MAXKC 8
#define MAXROUNDS 14

typedef unsigned char word8;
typedef unsigned int word32;

int BC, KC, ROUNDS;
int s;

word8 Logtable[256] = {
            0,   0, 25,   1, 50,   2, 26,198, 75,199, 27,104, 51,238,223,   3,
          100,   4,224, 14, 52,141,129,239, 76,113,   8,200,248,105, 28,193,
          125,194, 29,181,249,185, 39,106, 77,228,166,114,154,201,   9,120,
          101, 47,138,   5, 33, 15,225, 36, 18,240,130, 69, 53,147,218,142,
          150,143,219,189, 54,208,206,148, 19, 92,210,241, 64, 70,131, 56,
          102,221,253, 48,191,   6,139, 98,179, 37,226,152, 34,136,145, 16,
          126,110, 72,195,163,182, 30, 66, 58,107, 40, 84,250,133, 61,186,
           43,121, 10, 21,155,159, 94,202, 78,212,172,229,243,115,167, 87,
          175, 88,168, 80,244,234,214,116, 79,174,233,213,231,230,173,232,
           44,215,117,122,235, 22, 11,245, 89,203, 95,176,156,169, 81,160,
          127, 12,246,111, 23,196, 73,236,216, 67, 31, 45,164,118,123,183,
          204,187, 62, 90,251, 96,177,134, 59, 82,161,108,170, 85, 41,157,
          151,178,135,144, 97,190,220,252,188,149,207,205, 55, 63, 91,209,
           83, 57,132, 60, 65,162,109, 71, 20, 42,158, 93, 86,242,211,171,
           68, 17,146,217, 35, 32, 46,137,180,124,184, 38,119,153,227,165,
          103, 74,237,222,197, 49,254, 24, 13, 99,140,128,192,247,112,   7};

word8 Alogtable[256] = {
            1,   3,   5, 15, 17, 51, 85,255, 26, 46,114,150,161,248, 19, 53,
           95,225, 56, 72,216,115,149,164,247,   2,   6, 10, 30, 34,102,170,
          229, 52, 92,228, 55, 89,235, 38,106,190,217,112,144,171,230, 49,
           83,245,   4, 12, 20, 60, 68,204, 79,209,104,184,211,110,178,205,
           76,212,103,169,224, 59, 77,215, 98,166,241,   8, 24, 40,120,136,
          131,158,185,208,107,189,220,127,129,152,179,206, 73,219,118,154,
          181,196, 87,249, 16, 48, 80,240, 11, 29, 39,105,187,214, 97,163,
          254, 25, 43,125,135,146,173,236, 47,113,147,174,233, 32, 96,160,
          251, 22, 58, 78,210,109,183,194, 93,231, 50, 86,250, 21, 63, 65,
          195, 94,226, 61, 71,201, 64,192, 91,237, 44,116,156,191,218,117,
          159,186,213,100,172,239, 42,126,130,157,188,223,122,142,137,128,
          155,182,193, 88,232, 35,101,175,234, 37,111,177,200, 67,197, 84,
          252, 31, 33, 99,165,244,   7,   9, 27, 45,119,153,176,203, 70,202,
           69,207, 74,222,121,139,134,145,168,227, 62, 66,198, 81,243, 14,
           18, 54, 90,238, 41,123,141,140,143,138,133,148,167,242, 13, 23,
           57, 75,221,124,132,151,162,253, 28, 36,108,180,199, 82,246,   1};

word8 S[256] = {
           99,124,119,123,242,107,111,197, 48,   1,103, 43,254,215,171,118,
          202,130,201,125,250, 89, 71,240,173,212,162,175,156,164,114,192,
          183,253,147, 38, 54, 63,247,204, 52,165,229,241,113,216, 49, 21,
```

```
            4,199, 35,195, 24,150,   5,154,   7, 18,128,226,235, 39,178,117,
            9,131, 44, 26, 27,110, 90,160, 82, 59,214,179, 41,227, 47,132,
           83,209,   0,237, 32,252,177, 91,106,203,190, 57, 74, 76, 88,207,
          208,239,170,251, 67, 77, 51,133, 69,249,   2,127, 80, 60,159,168,
           81,163, 64,143,146,157, 56,245,188,182,218, 33, 16,255,243,210,
          205, 12, 19,236, 95,151, 68, 23,196,167,126, 61,100, 93, 25,115,
           96,129, 79,220, 34, 42,144,136, 70,238,184, 20,222, 94, 11,219,
          224, 50, 58, 10, 73,   6, 36, 92,194,211,172, 98,145,149,228,121,
          231,200, 55,109,141,213, 78,169,108, 86,244,234,101,122,174,   8,
          186,120, 37, 46, 28,166,180,198,232,221,116, 31, 75,189,139,138,
          112, 62,181,102, 72,   3,246, 14, 97, 53, 87,185,134,193, 29,158,
          225,248,152, 17,105,217,142,148,155, 30,135,233,206, 85, 40,223,
          140,161,137, 13,191,230, 66,104, 65,153, 45, 15,176, 84,187, 22};

word8 Si[256] = {
            82,   9,106,213, 48, 54,165, 56,191, 64,163,158,129,243,215,251,
          124,227, 57,130,155, 47,255,135, 52,142, 67, 68,196,222,233,203,
           84,123,148, 50,166,194, 35, 61,238, 76,149, 11, 66,250,195, 78,
            8, 46,161,102, 40,217, 36,178,118, 91,162, 73,109,139,209, 37,
          114,248,246,100,134,104,152, 22,212,164, 92,204, 93,101,182,146,
          108,112, 72, 80,253,237,185,218, 94, 21, 70, 87,167,141,157,132,
          144,216,171,   0,140,188,211, 10,247,228, 88,   5,184,179, 69,   6,
          208, 44, 30,143,202, 63, 15,   2,193,175,189,   3,   1, 19,138,107,
           58,145, 17, 65, 79,103,220,234,151,242,207,206,240,180,230,115,
          150,172,116, 34,231,173, 53,133,226,249, 55,232, 28,117,223,110,
           71,241, 26,113, 29, 41,197,137,111,183, 98, 14,170, 24,190, 27,
          252, 86, 62, 75,198,210,121, 32,154,219,192,254,120,205, 90,244,
           31,221,168, 51,136,   7,199, 49,177, 18, 16, 89, 39,128,236, 95,
           96, 81,127,169, 25,181, 74, 13, 45,229,122,159,147,201,156,239,
          160,224, 59, 77,174, 42,245,176,200,235,187, 60,131, 83,153, 97,
           23, 43,   4,126,186,119,214, 38,225,105, 20, 99, 85, 33, 12,125};


word32 RC[30] = {
          0x00,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,
          0x1b,0x36,0x6c,0xd8,0xab,0x4d,0x9a,0x2f,0x5e,
          0xbc,0x63,0xc6,0x97,0x35,0x6a,0xd4,0xb3,0x7d,
          0xfa,0xef,0xc5};



static word8 shifts[5][4] = {
          0,1,2,3,
          0,1,2,3,
          0,1,2,3,
          0,1,2,4,
          0,1,3,4};

static int numrounds[5][5] = {
          10,11,12,13,14,
          11,11,12,13,14,
          12,12,12,13,14,
          13,13,13,13,14,
          14,14,14,14,14};



word8 mul(word8 a, word8 b) {
          if(a && b) return Alogtable[(Logtable[a] + Logtable[b])%255];
```

```
        else return 0;
}

void AddRoundKey(word8 a[4][MAXBC], word8 rk[4][MAXBC]) {
        int i, j;

        for(i = 0; i<4; i++)
                for(j=0;j<BC;j++) a[i][j] ^= rk[i][j];
}

void SubBytes(word8 a[4][MAXBC], word8 box[256]){
        int i,j;

        for(i=0;i<4;i++)
                for(j=0;j<BC;j++) a[i][j] = box[a[i][j]] ;
}

void ShiftRows(word8 a[4][MAXBC], word8 d){
        word8 tmp[MAXBC];
        int i,j;

        if(d==0){
                for(i=1;i<4;i++){
                        for(j=0;j<BC;j++)
                                tmp[j] = a[i][(j+shifts[BC-4][i]) % BC];
                        for(j=0;j<BC;j++) a[i][j] = tmp[j];
                }
        }
        else{
                for(i=1;i<4;i++){
                        for(j=0;j<BC;j++)
                                tmp[j] = a[i][(BC+j-shifts[BC-4][i]) % BC];
                        for(j=0;j<BC;j++) a[i][j] = tmp[j];
                }
        }
}

void MixColumns(word8 a[4][MAXBC]){
        word8 b[4][MAXBC];
        int i,j;

        for(j=0;j<BC;j++)
                for(i=0;i<4;i++)
                        b[i][j] = mul(2,a[i][j])
                                ^ mul(3,a[(i+1) % 4][j])
                                ^ a[(i+2) % 4][j]
                                ^ a[(i+3) % 4][j];
        for(i=0;i<4;i++)
                for(j=0;j<BC;j++) a[i][j] = b[i][j];
}

void InvMixColumns(word8 a[4][MAXBC]){
        word8 b[4][MAXBC];
        int i,j;

        for(j=0;j<BC;j++)
        for(i=0;i<4;i++)
                b[i][j] = mul(0xe, a[i][j])
                        ^ mul(0xb, a[(i+1) % 4][j])
```

```
                        ^ mul(0xd, a[(i+2) % 4][j])
                        ^ mul(0x9, a[(i+3) % 4][j]);
        for(i=0;i<4;i++)
                for(j=0;j<BC;j++) a[i][j] = b[i][j];
}

int KeyExpansion(word8 k[4][MAXKC],
                                word8 W[MAXROUNDS+1][4][MAXBC]){
        int i,j,t,RCpointer = 1;
        word8 tk[4][MAXKC];

        for(j=0;j<KC;j++)
                for(i=0;i<4;i++)
                        tk[i][j] = k[i][j];
        t = 0;

        for(j=0;(j<KC) && (t<(ROUNDS+1)*BC); j++, t++)
                for(i=0;i<4;i++) W[t / BC][i][t % BC] = tk[i][j];

        while(t<(ROUNDS+1)*BC){
                for(i=0;i<4;i++)
                        tk[i][0] ^= S[tk[(i+1)%4][KC-1]];
                tk[0][0] ^= RC[RCpointer++];

                if(KC<=6)
                        for(j=1;j<KC;j++)
                                for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
                else{
                        for(j=1;j<4;j++)
                                for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
                        for(i=0;i<4;i++) tk[i][4] ^= S[tk[i][3]];
                        for(j=5;j<KC;j++)
                                for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
                }
                for(j=0; (j<KC) && (t<(ROUNDS+1)*BC); j++, t++)
                        for(i=0;i<4;i++) W[t/BC][i][t%BC] = tk[i][j];
        }
        return 0;
}
/*
int Encrypt(word8 a[4][MAXBC], word8 rk[MAXROUNDS+1][4][MAXBC]){
        int r;

        AddRoundKey(a,rk[0]);

        for(r=1;r<ROUNDS;r++){
                SubBytes(a,S);
                ShiftRows(a,0);
                MixColumns(a);
                AddRoundKey(a,rk[r]);
        }

        SubBytes(a,S);
        ShiftRows(a,0);
        AddRoundKey(a,rk[ROUNDS]);

        return 0;
}
*/
```

```
int Decrypt(word8 a[4][MAXBC], word8 rk[MAXROUNDS+1][4][MAXBC]){
	int r;

	AddRoundKey(a,rk[ROUNDS]);
	SubBytes(a,Si);
	ShiftRows(a,1);

	for(r=ROUNDS-1;r>0;r--){
		AddRoundKey(a,rk[r]);
		InvMixColumns(a);
		SubBytes(a,Si);
		ShiftRows(a,1);
	}
	AddRoundKey(a,rk[0]);

	return 0;
}


int main(int argc, char* argv[])
{
	int i,j,klen,blen;
	word8 a[4][MAXBC], rk[MAXROUNDS+1][4][MAXBC], sk[4][MAXKC];
	char c_klen[8], c_blen[8], pass[128];
	FILE *fkey =0, *fin = 0, *fout = 0;
    fpos_t flen;
	char *dbuf;
    unsigned long len, rlen, blen4;

////////////////////////////////////////////////////////////
	printf("Start!¥n");

	if( argc != 4 ){                                    // 使い方の誤り
		printf( "argc != 4 ¥n" );
		return -1;
	}

	if((fkey = fopen(argv[1], "rt")) == NULL){
		printf("Can not find key file for decryption. ¥n");
		return (-1);
	}
	fgets( c_klen, 6, fkey );
	fgets( c_blen, 6, fkey );
	fgets( pass, 127, fkey );

	klen = atoi(c_klen)/32;
	blen = atoi(c_blen)/32;
	blen4 = (unsigned int)blen*4;


	dbuf = (char *)malloc(2*blen4);


	KC = klen;
	if(KC<4 || 8<KC){
		printf("Wrong key size. ¥n");
		return (-1);
	}
```

```
        BC = blen;
        if(BC<4 || 8<BC){
                printf("Wrong block size. ¥n");
                return (-1);
        }


        if((fin = fopen(argv[2], "rb")) == NULL){
                printf("Can not open encrypted file. ¥n");
                return (-1);
        }
    fseek(fin, 0, SEEK_END);
    fgetpos(fin, &flen);
    rlen = file_len(flen);
    // reset to start
    fseek(fin, 0, SEEK_SET);

        if((fout = fopen(argv[3], "wb")) == NULL){
                printf("Can not open decrypted file. ¥n");
                return (-1);
        }

        ROUNDS = numrounds[KC-4][BC-4];

        int k = 0;
        for(j=0;j<KC;j++){
                for(i=0;i<4;i++){
                        sk[i][j] = pass[k];
                        k += 1;
                }
        }
        KeyExpansion(sk,rk);
        s = sizeof(unsigned int);
//////////////////////////////

        len = (unsigned long) fread(dbuf, 1, blen4 , fin);
    rlen -= len;

    // decrypt the top 16 bytes of the buffer
        k=0;
        for(j=0;j<BC;j++){
                for(i=0;i<4;i++){
                        a[i][j] = dbuf[k];
                        k++;
                }
        }

        Decrypt(a,rk);
    len = blen4 ;

        // write the IV and the encrypted file bytes
        k=0;
        for(j=0;j<BC;j++){
                for(i=0;i<4;i++){
                        dbuf[k] = a[i][j];
                        k++;
                }
        }
```

```
        unsigned int wlen = *((unsigned int*)dbuf);

        if(wlen <= len-s){
        if(fwrite(dbuf+s, 1, wlen, fout) != wlen)
                return -2;
        }
        else{
                wlen -= len-s;
        if(fwrite(dbuf+s, 1, len-s, fout) != len-s)
                return -2;
        }

if((rlen <= blen4) && (rlen>0) )
{     // if the original file length is less than or equal to 16 bytes
        // read the bytes of the file and verify length
        len = (unsigned long)fread(dbuf, 1, blen4 , fin);
        rlen -= len;

                if(rlen > 0)
            return -1;

         // decrypt from position len to position len + BLOCK_LEN
                k=0;
                for(j=0;j<BC;j++){
                        for(i=0;i<4;i++){
                                a[i][j] = dbuf[k];
                                k++;
                        }
                }

                Decrypt(a,rk);

                k=0;
                for(j=0;j<BC;j++){
                        for(i=0;i<4;i++){
                                dbuf[k] = a[i][j];
                                k++;
                        }
                }

        // output decrypted bytes
        if(fwrite(dbuf, 1, wlen, fout) != (unsigned long)wlen)
            return -2;
}
else
{
        // read the encrypted file a block at a time
        while(rlen > 0 && !feof(fin))
        {
            // input a block and reduce the remaining byte count
            len = (unsigned long)fread(dbuf, 1, blen4, fin);
            rlen -= len;

            // verify the length of the read operation
            if(len != blen4 )
                return -1;

            // decrypt input buffer
                        k=0;
```

```
                        for(j=0;j<BC;j++){
                                for(i=0;i<4;i++){
                                        a[i][j] = dbuf[k];
                                        k++;
                                }
                        }

                        Decrypt(a,rk);

                        k=0;
                        for(j=0;j<BC;j++){
                                for(i=0;i<4;i++){
                                        dbuf[k] = a[i][j];
                                        k++;
                                }
                        }

                        if(wlen < len){
                                if(fwrite(dbuf, 1, wlen, fout) != wlen){
                                    return -2;
                                }
                                break;
                        }
                        else{
                                if(fwrite(dbuf, 1, blen4, fout) != blen4)
                                        return -2;
                        }
                        if(wlen > len){wlen -= len;}
                }
        }

        if(fkey)
        {
        fclose(fkey);
        }

        if(fout)
        {
        fclose(fout);
        }

        if(fin)
        {
                fclose(fin);
        }

        free(dbuf);
////////////////////////////////////////////////
        printf("End!¥n");
        return 0;
}


//////////////////////////////////////////////////////////////////

// stdafx.cpp：標準インクルードを含むソース ファイルです。
// Crytest.pch は、プリコンパイル済みヘッダーになります。
// stdafx.obj にはプリコンパイル済み方情報が含まれます。
```

```
#include "stdafx.h"


////////////////////////////////////////////////////////////////////////

// stdafx.h : 標準のシステム インクルード ファイルのインクルード ファイル、または
// 参照回数が多く、かつあまり変更されない、プロジェクト専用のインクルード ファイル
// を記述します。

#pragma once

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN            // Windows ヘッダーから使用されていない部分を除外します。
#endif

// 下で指定された定義の前に対象プラットフォームを指定しなければならない場合、以下の定義を変更してください。
// 異なるプラットフォームに対応する値に関する最新情報については、MSDN を参照してください。
#ifndef WINVER                          // Windows 95 および Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define WINVER 0x0400           // これを Windows 98 および Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINNT            // Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINNT 0x0400             // これを Windows 98 および Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINDOWS                  // Windows 98 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINDOWS 0x0500 // これを Windows Me またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_IE                       // IE 4.0 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_IE 0x0400        // これを IE 5.0  またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS      // 一部の CString コンストラクタは明示的です。

// 一般的で無視しても安全な MFC の警告メッセージの一部の非表示を解除します。
#define _AFX_ALL_WARNINGS

#include <afxwin.h>             // MFC のコアおよび標準コンポーネント
#include <afxext.h>             // MFC の拡張部分

#include <afxdtctl.h>                   // MFC の Internet Explorer 4 コモン コントロール サポート
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC の Windows コモン コントロール サポート
#endif // _AFX_NO_AFXCMN_SUPPORT
```

１０．２．３　鍵の形式

鍵は次のような 3 行からなるテキストファイルです。

```
128
128
40B6E45ED3FE4CA2D1E14EDD9C48FC68
```

1 行目は、鍵の長さ
2 行目は、ブロックサイズ
3 行目は鍵を 16 進数で表したものです。

１０．３　Twofish 暗号ソースコード

参考文献："The Twofish Encryption Algorithm",　(WILEY)に記載されているものとほぼ同じですが、製作過程で内容がわかりやすくなるようにするために新たな変数を使用しました。これについては皆様自信で参考文献のソースコードと比較されるのが良いと思います。また、

Twofish is unpatented, and the source code is uncopyrighted and license-free; it is free for all uses

となっています。

１０．３．１　暗号化部分

```
// ECTwofish.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include"aes.h"
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include<ctype.h>

extern   CONST char *moduleDescription; /* which module is running */
extern   CONST char *modeString;                              /* which key schedule mode */
extern   CONST int   debugCompile;                  /* is external module compiled with debug?
*/

#if defined(__WATCOMC__) && defined(_M_IX86) && !defined(NO_TIMER)
     DWORD ReadTimeStampCounter(void);
          #pragma aux ReadTimeStampCounter = " db 0Fh,031h" value [eax] modify exact [eax
edx] // RDTSC opcode
#endif


/*
+*****************************************************************************
*                              Constants/Macros/Tables
-*****************************************************************************/

typedef struct
          {
          FILE *f;                              /* the file being written/read */
          int   I;                                        /* test number */
          int         keySize;                            /* key size in bits */
          int         gotDebugIO;               /* got any debug IO? */
          BYTE pt[BLOCK_SIZE/8];          /* plaintext */
          BYTE ct[BLOCK_SIZE/8];          /* ciphertext */
```

```
        keyInstance     ki;             /* use ki.keyDwords as key bits */
        cipherInstance ci;              /* use ci.iv as iv bits */
        } testData;


static char hexTab[]    =       "0123456789ABCDEF";
char            filePath[80]=   "";

int                     useAsm          =       0;      /* use assembly language */
int                     mctInner        =       MCT_INNER/100;
int                     mctOuter        =       MCT_OUTER/10;
int                     verify          =       0;      /* set to nonzero to read&verify
files */
int                     debug           =       0;      /* debugging mode */
int                     verbose         =       0;      /* verbose output */
int                     quietVerify     =       0;      /* quiet during verify */
int                     timeIterCnt     =       0;      /* how many times to iterate for
timing */
DWORD           randBits[64]= {1};      /* use Knuth's additive generator */
int                     randPtr;
testData *      debugTD                 = NULL;         /* for use with debugIO */
int                     CLKS_BYTE       =       0;      /* use clks/byte? (vs. clks/block) */
int                     FMT_LOG                 =       0;      /* format for log file */
int                     CLK_MHZ                 =       200;/* default clock speed */

#define         KEY_BITS_0                      128                     /* first key bit
setting to test */
#define         STEP_KEY_BITS                   ((MAX_KEY_BITS-KEY_BITS_0)/2)
/*
static char   hexString[]=

        "0123456789ABCDEFFEDCBA98765432100011223344556677889AABBCCDDEEFF"
;
*/
static char   hex7String[72];
//=
        "12345671234567123456712345671234567123456712345671234567123456712345671";
////////////////////////////////////////////
/*
+*****************************************************************************
*                               Functions
-*****************************************************************************/

DWORD Here(DWORD x)
        {
        unsigned int mask=~0U;

        return (* (((DWORD *)&x)-1)) & mask;
        }
extern DWORD TwofishCodeSize(void);

#ifdef USE_ASM
int cdecl get_cpu_type(void);                   /* return CPU type */
#endif

/*
+*****************************************************************************
*
```

```
* Function Name:           Rand
*
* Function:                      Generate random number
*
* Arguments:                None.
*
* Return:                        New random number.
*
* Notes:                    Uses Knuth's additive generator, other magic
*
-*************************************************************************/
DWORD Rand(void)
        {
        if (randPtr >= 57)
                randPtr = 0;                            /* handle the ptr wrap */

        randBits[randPtr] += randBits[(randPtr < 7) ? randPtr-7+57 : randPtr-7];

        randBits[62]+= randBits[61];
        randBits[63] = ROL(randBits[63],9) + 0x6F4ED7D0;/* very long period! */

        return (randBits[randPtr++] ^ randBits[63]) + randBits[62];
        }


/*
+*************************************************************************
*
* Function Name:           SetRand
*
* Function:                      Initialize random number seed
*
* Arguments:                seed    =       new seed value
*
* Return:                        None.
*
* Notes:
*
-*************************************************************************/
void SetRand(DWORD seed)
        {
        int i;
        DWORD x;

        randPtr=0;
        for (i=x=0;i<64;i++)
                {
                randBits[i]=seed;
                x |= seed;                      /* keep track of lsb of all entries */
                seed = ROL(seed,11) + 0x12345678;
                }

        if ((x & 1) == 0)    /* insure maximal period by having at least one odd value */
                randBits[0]++;

        for (i=0;i<1000;i++)
                Rand();                         /* run it for a while */

        randBits[63] = Rand();
```

```
        randBits[62] = Rand();
        randBits[61] = Rand() | 1;/* make it odd */
        }


/*
+*****************************************************************************
*
* Function Name:        ClearTestData
*
* Function:                     Initialize test data to all zeroes
*
* Arguments:            t               =       pointer to testData structure
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void ClearTestData(testData *t)
        {
        t->gotDebugIO=0;
        memset(t->pt,0,BLOCK_SIZE/8);
        memset(t->ct,0,BLOCK_SIZE/8);
        memset(t->ci.iv32,0,BLOCK_SIZE/8);
        memset(t->ki.key32,0,MAX_KEY_BITS/8);
        memset(t->ki.keyMaterial,'0',sizeof(t->ki.keyMaterial));
#if defined(COMPILE_KEY) && defined(USE_ASM)
        t->ki.cSig1=t->ki.cSig2=0;
#endif
        }

/*
+*****************************************************************************
*
* Function Name:        FatalError
*
* Function:                     Output a fatal error message and exit
*
* Arguments:            msg             =       fatal error description (printf string)
*                               msg2    =       2nd parameter to printf msg
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void FatalError(CONST char *msg,CONST char *msg2)
        {
        printf("¥nFATAL ERROR: ");
        printf(msg,msg2);
        exit(2);
        }


/*
+*****************************************************************************
*
* Function Name:        AES_FileIO
```

```
*
* Function:                     Output to file or verify file contents vs. string
*
* Arguments:        f           =       opened file
*                                       s               =       string to output/compare
(NULL-->reset, return)
*                                       errOK   =       do not fatalError on miscompare
*
* Return:                       Zero --> compare ok
*
* Notes:                        On miscompare, FatalError (unless errOK)
*
-*****************************************************************************/
int AES_FileIO(FILE *f,CONST char *s,int errOK)
        {
        int    i;
        static int    lineNum=0;
        static int    j=0;
        static char line[516]="";

        if (s == NULL)    /* starting new file */
                {
                line[0]=j=lineNum=0;
                return 0;
                }

        if (!verify)
                {
                fprintf(f,s);
                return 0;
                }

        /* here to verify the file against the string */
        for (i=0;s[i];i++)
                {
                while (line[j] == 0)
                        {
                        lineNum++;
                        if (fgets(line,sizeof(line)-4,f) == NULL)
                                {
                                if ((s[i]=='¥n') && (s[i+1]==0))
                                        {
                                        line[0]=j=0;        /* missing final eol is ok */
                                        return 0;
                                        }
                                FatalError("Unexpected EOF looking for %s",s);
                                }
                        if (verbose) printf(line);
                        j=0;
                        }
                if (s[i] != line[j])
                        {
                        if ((s[i] == '¥n') && ((i==0) || (s[i-1] == '¥n'))) continue; /* blank line
skip */
                        if (line[j] == '¥n') {j++; continue;}
                        if (!errOK)
                                {
                                char tmp[1024];
                                sprintf(tmp,"Miscompare    at    line    #%d:¥n%s¥nlooking
```

```
for¥n¥n%%s",lineNum,line);
                                        FatalError(tmp,s);
                                        }
                                line[0]=j=0;        /* let caller re-synch if desired */
                                return 1;                       /* return error flag */
                                }
                        j++;
                        }

        return 0;
        }



/*
+*****************************************************************************
*
* Function Name:        DebugIO
*
* Function:                     Output debug string
*
* Arguments:            s       =       string to output
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void DebugIO(CONST char *s)
        {
        if (debugTD)
                {
                AES_FileIO(debugTD->f,s,0);
                debugTD->gotDebugIO=1;
                }
        else
                printf(s);
        }


//////////////////////////////////////////////////////////////////
/*************************************************************/
////////////   uyama
void UY_EC_TF(char *key, char *pt, char *ct){
        FILE *fkey;
        FILE *stream1;
        FILE *stream2;
        char c_mode[8], c_klen[8], pass[128];
        int i,mode,klen;
        testData t;
        int c, block;
        long mesLength; // 平文長（バイト）
        char* bufp;
        unsigned char cstr[BLOCK_SIZE/8+10];
        // 暗号文格納場所へのポインタ

        printf("TfEC Start!¥n" );
        /*  鍵を読み出すファイルを開く*/
```

```
        if( (fkey   = fopen( key, "rt" )) == NULL ){
                printf( "Can not open key file.¥n");
                return;
        }
        fgets( c_mode, 6, fkey );
        fgets( c_klen, 6, fkey );
        fgets( pass, 127, fkey );
        mode = atoi(c_mode);
        klen = atoi(c_klen);
    /*  鍵*/
        for(i=0; i<klen/4; i++){
                hex7String[i] = pass[i];
        }
        hex7String[klen/4] = NULL;

        /*  平文を読み出すファイルを開く*/
        if( (stream1    = fopen( pt, "rb" )) == NULL )
                printf( "Can not open plane text file.¥n");
   /*  暗号文を書き込むファイルを開く  */
        if( (stream2 = fopen( ct ,"wb" )) == NULL )
                printf( "Can not open file for encrypted data.¥n" );

////////////////////////////////////////////////////////////////////
    //  平文
        fseek(stream1, 0, SEEK_END);
        long filelen = ftell(stream1);
        fseek(stream1,0,0);
        int head = sizeof(long);
        mesLength = filelen + head;

        if (cipherInit(&t.ci,mode,hex7String) != TRUE)
                FatalError("cipherInit error during %s test",""/*fname*/);
        t.keySize=klen;
        ClearTestData(&t);                                      /* start with all zeroes */
        if (makeKey(&t.ki,DIR_ENCRYPT,t.keySize,hex7String/*t.ki.keyMaterial*/) != TRUE)
                FatalError("Error parsing key during %s test",""/*fname*/);

        //暗号化
        if(mesLength <= BLOCK_SIZE/8){//63 が暗号化の作業サイズ   63*20＝1260
                bufp = (char*)new(char[BLOCK_SIZE/8+10]);
                if(bufp == NULL){
                        printf("No memory");
                        return;
                }
  //  平文
                *(long *)bufp = filelen;
                int i = head;
                do{
                        c = fgetc(stream1);
                        bufp[i]=c;
                        i=i+1;
                }while(c!=EOF);
                bufp[i-1]=NULL;
                for(int j=0; j+i<BLOCK_SIZE/8+10; j++){
                        bufp[j+i] = NULL;
                }
                mesLength = i-1; //  平文長（バイト) + head
                memcpy(t.pt,bufp,BLOCK_SIZE/8);
    //  暗号化実行
```

```
		if (blockEncrypt(&t.ci,&t.ki,t.pt,BLOCK_SIZE,t.ct) != BLOCK_SIZE)
			FatalError("blockEncrypt return during %s test","ff.bin"/*fname*/);
		memcpy(cstr,t.ct,BLOCK_SIZE/8);
		fseek(stream2,0,0);
		fwrite( cstr,sizeof(char),BLOCK_SIZE/8, stream2);
	}
	else{
		long rBlen = mesLength;
		bufp = (char*)new(char[BLOCK_SIZE/8]);//63 が暗号化の作業サイズ　63*20＝
1260
		if(bufp == NULL){
			printf("メモリ不足¥r¥n");
			return;
		}
		(*(long*)(bufp)) = filelen;

		int r = 0;
		do{
		// 平文
			if(r == 0){
				i = head;
				fseek(stream1, r*BLOCK_SIZE/8, 0);//63 が暗号化の作業サイ
ズ　63*20＝1260
			}
			if(r > 0 ){
				i = 0;
				fseek(stream1, r*BLOCK_SIZE/8-4, 0);//63 が暗号化の作業サ
イズ　63*20＝1260
			}
			do{
				c = fgetc(stream1);
				bufp[i]=c;
				i=i+1;
			}while((c!=EOF) && (i<=BLOCK_SIZE/8));//63 が暗号化の作業サイズ
63*20＝1260
			bufp[i-1]=NULL;
			for(int j=0; j+i<BLOCK_SIZE/8; j++){
				bufp[j+i] = NULL;
			}

			if(rBlen >= BLOCK_SIZE/8){ block = BLOCK_SIZE/8; }//63 が暗号化の
作業サイズ　63*20＝1260
			if(rBlen < BLOCK_SIZE/8){ block = rBlen;}

			memcpy(t.pt,bufp,BLOCK_SIZE/8);
	// 暗号化実行
			if (blockEncrypt(&t.ci,&t.ki,t.pt,BLOCK_SIZE,t.ct) != BLOCK_SIZE)
				FatalError("blockEncrypt return during %s
test","ff.bin"/*fname*/);

		// 暗号文を書き込む
			memcpy(cstr,t.ct,BLOCK_SIZE/8);
			fwrite( cstr,sizeof(char),BLOCK_SIZE/8, stream2);
			r += 1;
			rBlen -= block;
		}while(rBlen>0);
	}
	_fcloseall();
```

```
        printf("TfEC End!¥n" );
}

////////////////////////////////////////////////////////////////

/*****************************************************************/


/*
+*****************************************************************************
*
* Function Name:        GiveHelp
*
* Function:                     Print out list of command line switches
*
* Arguments:            None.
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void GiveHelp(void)
        {
        printf("Syntax:    TST2FISH [options]¥n"
                "Purpose:  Generate/validate AES Twofish code and files¥n"
                "Options:  -lNN     ==> set sanity check loop to NN¥n"
                "          -m       ==> do full MCT generation¥n"
                "          -pPath   ==> set file path¥n"
                "          -s       ==> set initial random seed based on time¥n"
                "          -sNN     ==> set initial random seed to NN¥n"
                "          -tNN     ==> time performance using NN iterations¥n"
                "          -v       ==> validate files, don't generate them¥n",
                MAX_ROUNDS
               );
        exit(1);
        }

#ifdef TEST_EXTERN
void    Test_Extern(void);
#endif

void ShowHex(FILE *f,CONST void *p,int bCnt,CONST char *name)
        {
        int i;

        fprintf(f,"     ;%s:",name);
        for (i=0;i<bCnt;i++)
                {
                if ((i % 8) == 0)
                        fprintf(f,"¥n¥t.byte¥t");
                else
                        fprintf(f,",");
                fprintf(f,"0%02Xh",((BYTE *)p)[i]);
                }
        fprintf(f,"¥n");
        }

/* output a formatted 6805 test vector include file */
```

```
void Debug6805(void)
	{
	int i,j;
	testData t;
	FILE *f;

	ClearTestData(&t);
	t.keySize=128;

	f=stdout;
	cipherInit(&t.ci,MODE_ECB,NULL);
	makeKey(&t.ki,DIR_ENCRYPT,t.keySize,t.ki.keyMaterial);

	for (i=0;i<4;i++)   /* make sure it all fits in 256 bytes */
		{
		reKey(&t.ki);
		blockEncrypt(&t.ci,&t.ki,t.pt,BLOCK_SIZE,t.ct);
		fprintf(f,"; Twofish vector #%d¥n",i+1);
		ShowHex(f,&t.keySize,1,"Key Size");
		ShowHex(f,t.ki.key32,16,"Key");
		ShowHex(f,t.pt,BLOCK_SIZE/8,"Plaintext");
		ShowHex(f,t.ct,BLOCK_SIZE/8,"Ciphertext");
		for (j=0;j<16;j++)
			((BYTE *)t.ki.key32)[j] = t.pt[j] ^ t.ct[j];
		memcpy(t.pt,t.ct,sizeof(t.pt));
		fprintf(f,";-----------------------------------------------------¥n");
		}
	fprintf(f,"¥n¥t.byte 0¥t;end of list¥n");
	fclose(f);
	}


////////////////////////////////////////////
int main(int argc, char* argv[])
{
#define  MAX_ARGS        40
	int i,testCnt=32;
	DWORD randSeed=0x12345678;
	char *moduleName=moduleDescription;

	// 引数チェック
	if( argc != 4 ){                                    // 使い方の誤り
		exit(1);
	}

	i=1;                            /* make sure LittleEndian is defined correctly */
	if (b0(i) != 1)
		FatalError("LittleEndian defined incorrectly","");
//	if ((ALIGN32) && (k == 2))
//		FatalError("Cannot enable ALIGN32 in 16-bit mode¥n","");

#if ((MCT_INNER != 10000) || (MCT_OUTER != 400))
#error   MCT loop counts incorrect!
#endif

#ifdef USE_ASM
	if (useAsm & 7)   moduleName="Assembler ";
#endif
```

```
	SetRand(randSeed);                                  /*  init  pseudorandom
generator for testing */

#ifdef TEST_EXTERN
	Test_Extern();
	exit(0);
#endif

	UY_EC_TF(argv[1],argv[2],argv[3]);

	return 0;
	}



//////////////////////////////////////////////////////

// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//		ECTwofish.pch 生成されるプリコンパイル済ヘッダー
//		stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。

//////////////////////////////////////////////////////////////

/*****************************************************************************
	TWOFISH2.C	-- Optimized C API calls for TWOFISH AES submission

	Submitters:
		Bruce Schneier, Counterpane Systems
		Doug Whiting,	Hi/fn
		John Kelsey,	Counterpane Systems
		Chris Hall,		Counterpane Systems
		David Wagner,	UC Berkeley

	Code Author:		Doug Whiting,	Hi/fn

	Version  1.00		April 1998

	Copyright 1998, Hi/fn and Counterpane Systems.  All rights reserved.

	Notes:
		*	Optimized version
		*	Tab size is set to 4 characters in this file

*****************************************************************************/
#include "stdafx.h"

#include"aes.h"
#include"table.h"

#include<memory.h>
#include<assert.h>

#define GetCodeSize //Uyama
```

```
#if     defined(min_key)    && !defined(MIN_KEY)
#define  MIN_KEY                    1                                   /* toupper() */
#elif defined(part_key) && !defined(PART_KEY)
#define  PART_KEY        1
#elif defined(zero_key) && !defined(ZERO_KEY)
#define  ZERO_KEY        1
#endif


#ifdef USE_ASM
extern    int        useAsm;                                 /* ok to use ASM code? */

typedef  int cdecl CipherProc
    (cipherInstance *cipher, keyInstance *key,BYTE *input,int inputLen,BYTE *outBuffer);
typedef int         cdecl KeySetupProc(keyInstance *key);

extern CipherProc           *blockEncrypt_86;           /* ptr to ASM functions */
extern CipherProc           *blockDecrypt_86;
extern KeySetupProc         *reKey_86;
extern DWORD                cdecl TwofishAsmCodeSize(void);
#endif

/*
+*****************************************************************************
*                           Constants/Macros/Tables
-*****************************************************************************/

#define             CONST                                           /* help syntax from C++, NOP here
*/

CONST               fullSbox MDStab;                    /* not actually const.    Initialized ONE time
*/
int                         needToBuildMDS=1;                   /* is MDStab initialized yet? */

#define             BIG_TAB                 0

#if BIG_TAB
BYTE                bigTab[4][256][256];        /* pre-computed S-box */
#endif

/* number of rounds for various key sizes:    128, 192, 256 */
/* (ignored for now in optimized code!) */
CONST int           numRounds[4]= {0,ROUNDS_128,ROUNDS_192,ROUNDS_256};

#if REENTRANT
#define             _sBox_     key->sBox8x32
#else
static              fullSbox _sBox_;            /* permuted MDStab based on keys */
#endif
#define _sBox8_(N) (((BYTE *) _sBox_) + (N)*256)

/*------- see what level of S-box precomputation we need to do -----*/
#if     defined(ZERO_KEY)
#define  MOD_STRING    "(Zero S-box keying)"
#define  Fe32_128(x,R)     ¥
          (         MDStab[0][p8(01)[p8(02)[_b(x,R    )]^b0(SKEY[1])]^b0(SKEY[0])] ^     ¥
                    MDStab[1][p8(11)[p8(12)[_b(x,R+1)]^b1(SKEY[1])]^b1(SKEY[0])] ^     ¥
                    MDStab[2][p8(21)[p8(22)[_b(x,R+2)]^b2(SKEY[1])]^b2(SKEY[0])] ^     ¥
                    MDStab[3][p8(31)[p8(32)[_b(x,R+3)]^b3(SKEY[1])]^b3(SKEY[0])] )
```

```
#define  Fe32_192(x,R)     ¥
         (
         MDStab[0][p8(01)[p8(02)[p8(03)[_b(x,R     )]^b0(SKEY[2])]^b0(SKEY[1])]^b0(SKEY[0])]
^ ¥

         MDStab[1][p8(11)[p8(12)[p8(13)[_b(x,R+1)]^b1(SKEY[2])]^b1(SKEY[1])]^b1(SKEY[0])]
^ ¥

         MDStab[2][p8(21)[p8(22)[p8(23)[_b(x,R+2)]^b2(SKEY[2])]^b2(SKEY[1])]^b2(SKEY[0])]
^ ¥

         MDStab[3][p8(31)[p8(32)[p8(33)[_b(x,R+3)]^b3(SKEY[2])]^b3(SKEY[1])]^b3(SKEY[0])]
)
#define  Fe32_256(x,R)     ¥
         (
         MDStab[0][p8(01)[p8(02)[p8(03)[p8(04)[_b(x,R     )]^b0(SKEY[3])]^b0(SKEY[2])]^b0(SK
EY[1])]^b0(SKEY[0])] ^ ¥

         MDStab[1][p8(11)[p8(12)[p8(13)[p8(14)[_b(x,R+1)]^b1(SKEY[3])]^b1(SKEY[2])]^b1(SK
EY[1])]^b1(SKEY[0])] ^ ¥

         MDStab[2][p8(21)[p8(22)[p8(23)[p8(24)[_b(x,R+2)]^b2(SKEY[3])]^b2(SKEY[2])]^b2(SK
EY[1])]^b2(SKEY[0])] ^ ¥

         MDStab[3][p8(31)[p8(32)[p8(33)[p8(34)[_b(x,R+3)]^b3(SKEY[3])]^b3(SKEY[2])]^b3(SK
EY[1])]^b3(SKEY[0])] )

#define  GetSboxKey       DWORD SKEY[4];            /* local copy */ ¥
                                                    memcpy(SKEY,key->sboxKeys,sizeof(SKEY));
/*-----------------------------------------------------------------*/
#elif defined(MIN_KEY)
#define  MOD_STRING    "(Minimal keying)"
#define  Fe32_(x,R)(MDStab[0][p8(01)[_sBox8_(0)[_b(x,R    )]] ^ b0(SKEY0)] ^ ¥
                                                    MDStab[1][p8(11)[_sBox8_(1)[_b(x,R+1)]] ^ b1(SKEY0)] ^
¥
                                                    MDStab[2][p8(21)[_sBox8_(2)[_b(x,R+2)]] ^ b2(SKEY0)] ^
¥
                                                    MDStab[3][p8(31)[_sBox8_(3)[_b(x,R+3)]] ^ b3(SKEY0)])
#define sbSet(N,i,J,v) { _sBox8_(N)[i+J] = v; }
#define  GetSboxKey       DWORD SKEY0 = key->sboxKeys[0]                    /* local copy */
/*-----------------------------------------------------------------*/
#elif defined(PART_KEY)
#define  MOD_STRING    "(Partial keying)"
#define  Fe32_(x,R)(MDStab[0][_sBox8_(0)[_b(x,R    )]] ^ ¥
                                                    MDStab[1][_sBox8_(1)[_b(x,R+1)]] ^ ¥
                                                    MDStab[2][_sBox8_(2)[_b(x,R+2)]] ^ ¥
                                                    MDStab[3][_sBox8_(3)[_b(x,R+3)]])
#define sbSet(N,i,J,v) { _sBox8_(N)[i+J] = v; }
#define  GetSboxKey
/*-----------------------------------------------------------------*/
#else     /* default is FULL_KEY */
#ifndef FULL_KEY
#define  FULL_KEY         1
#endif
#if BIG_TAB
#define  TAB_STR                    " (Big table)"
#else
#define  TAB_STR
#endif
```

```
#ifdef COMPILE_KEY
#define  MOD_STRING    "(Compiled subkeys)" TAB_STR
#else
#define  MOD_STRING    "(Full keying)" TAB_STR
#endif
/* Fe32_ does a full S-box + MDS lookup.    Need to #define _sBox_ before use.
    Note that we "interleave" 0,1, and 2,3 to avoid cache bank collisions
    in optimized assembly language.
*/
#define  Fe32_(x,R) (_sBox_[0][2*_b(x,R    )] ^ _sBox_[0][2*_b(x,R+1)+1] ^        ¥
                                        _sBox_[2][2*_b(x,R+2)] ^ _sBox_[2][2*_b(x,R+3)+1])
                    /* set a single S-box value, given the input byte */
#define sbSet(N,i,J,v) { _sBox_[N&2][2*i+(N&1)+2*J]=MDStab[N][v]; }
#define  GetSboxKey
#endif

CONST           char *moduleDescription  ="Optimized C ";
CONST           char *modeString                    =MOD_STRING;


/* macro(s) for debugging help */
#define         CHECK_TABLE             0               /*  nonzero  -->  compare  against
"slow" table */
#define         VALIDATE_PARMS          0               /* disable for full speed */

#include"debug.h"                                       /* debug display macros */

/* end of debug macros */

#ifdef GetCodeSize
extern DWORD Here(DWORD x);                             /* return caller's address! */
DWORD TwofishCodeStart(void) { return Here(0); }
#endif

/*
+*****************************************************************************
*
* Function Name:        TableOp
*
* Function:                     Handle table use checking
*
* Arguments:            op      =       what to do      (see TAB_* defns in AES.H)
*
* Return:                       TRUE --> done (for TAB_QUERY)
*
* Notes: This routine is for use in generating the tables KAT file.
*                       For this optimized version, we don't actually track table usage,
*                       since it would make the macros incredibly ugly.    Instead we just
*                       run for a fixed number of queries and then say we're done.
*
-*****************************************************************************/
int TableOp(int op)
        {
        static int queryCnt=0;

        switch (op)
                {
                case TAB_DISABLE:
                        break;
```

```
                case TAB_ENABLE:
                        break;
                case TAB_RESET:
                        queryCnt=0;
                        break;
                case TAB_QUERY:
                        queryCnt++;
                        if (queryCnt < TAB_MIN_QUERY)
                                return FALSE;
                }
        return TRUE;
        }


/*
+*************************************************************************
*
* Function Name:        ParseHexDword
*
* Function:                     Parse ASCII hex nibbles and fill in key/iv dwords
*
* Arguments:            bit                     =       # bits to read
*                                       srcTxt          =       ASCII source
*                                       d                       =       ptr to dwords to
fill in
*                                       dstTxt          =       where to make a copy of
ASCII source
*                                                                       (NULL ok)
*
* Return:                       Zero if no error.   Nonzero --> invalid hex or length
*
* Notes:   Note that the parameter d is a DWORD array, not a byte array.
*       This routine is coded to work both for little-endian and big-endian
*       architectures.   The character stream is interpreted as a LITTLE-ENDIAN
*       byte stream, since that is how the Pentium works, but the conversion
*       happens automatically below.
*
-*************************************************************************/
int ParseHexDword(int bits,CONST char *srcTxt,DWORD *d,char *dstTxt)
        {
        int i;
        char c;
        DWORD b;

        union   /* make sure LittleEndian is defined correctly */
                {
                BYTE   b[4];
                DWORD d[1];
                } v;
        v.d[0]=1;
        if (v.b[0 ^ ADDR_XOR] != 1)
                return BAD_ENDIAN;              /* make sure compile-time switch is set ok */

#if VALIDATE_PARMS
  #if ALIGN32
        if (((int)d) & 3)
                return BAD_ALIGN32;
  #endif
#endif
```

```
    for (i=0;i*32<bits;i++)
        d[i]=0;                                 /* first, zero the field */

    for (i=0;i*4<bits;i++)          /* parse one nibble at a time */
        {                                       /* case out the
hexadecimal characters */
        c=srcTxt[i];
        if (dstTxt) dstTxt[i]=c;
        if ((c >= '0') && (c <= '9'))
            b=c-'0';
        else if ((c >= 'a') && (c <= 'f'))
            b=c-'a'+10;
        else if ((c >= 'A') && (c <= 'F'))
            b=c-'A'+10;
        else
            return BAD_KEY_MAT;   /* invalid hex character */
        /* works for big and little endian! */
        d[i/8] |= b << (4*((i^1)&7));
        }

    return 0;                                   /* no error */
    }


#if CHECK_TABLE
/*
+*****************************************************************************
*
* Function Name:    f32
*
* Function:         Run four bytes through keyed S-boxes and apply MDS matrix
*
* Arguments:        x           =   input to f function
*                   k32         =   pointer to key
dwords
*                   keyLen      =   total key length (k32 -->
keyLey/2 bits)
*
* Return:           The output of the keyed permutation applied to x.
*
* Notes:
*   This function is a keyed 32-bit permutation.  It is the major building
*   block for the Twofish round function, including the four keyed 8x8
*   permutations and the 4x4 MDS matrix multiply.  This function is used
*   both for generating round subkeys and within the round function on the
*   block being encrypted.
*
*   This version is fairly slow and pedagogical, although a smartcard would
*   probably perform the operation exactly this way in firmware.  For
*   ultimate performance, the entire operation can be completed with four
*   lookups into four 256x32-bit tables, with three dword xors.
*
*   The MDS matrix is defined in TABLE.H.  To multiply by Mij, just use the
*   macro Mij(x).
*
-*****************************************************************************/
DWORD f32(DWORD x,CONST DWORD *k32,int keyLen)
    {
```

```
        BYTE    b[4];

        /* Run each byte thru 8x8 S-boxes, xoring with key byte at each stage. */
        /* Note that each byte goes through a different combination of S-boxes.*/

        *((DWORD *)b) = Bswap(x);           /* make b[0] = LSB, b[3] = MSB */
        switch (((keyLen + 63)/64) & 3)
                {
                case 0:                 /* 256 bits of key */
                        b[0] = p8(04)[b[0]] ^ b0(k32[3]);
                        b[1] = p8(14)[b[1]] ^ b1(k32[3]);
                        b[2] = p8(24)[b[2]] ^ b2(k32[3]);
                        b[3] = p8(34)[b[3]] ^ b3(k32[3]);
                        /* fall thru, having pre-processed b[0]..b[3] with k32[3] */
                case 3:                 /* 192 bits of key */
                        b[0] = p8(03)[b[0]] ^ b0(k32[2]);
                        b[1] = p8(13)[b[1]] ^ b1(k32[2]);
                        b[2] = p8(23)[b[2]] ^ b2(k32[2]);
                        b[3] = p8(33)[b[3]] ^ b3(k32[2]);
                        /* fall thru, having pre-processed b[0]..b[3] with k32[2] */
                case 2:                 /* 128 bits of key */
                        b[0] = p8(00)[p8(01)[p8(02)[b[0]] ^ b0(k32[1])] ^ b0(k32[0])];
                        b[1] = p8(10)[p8(11)[p8(12)[b[1]] ^ b1(k32[1])] ^ b1(k32[0])];
                        b[2] = p8(20)[p8(21)[p8(22)[b[2]] ^ b2(k32[1])] ^ b2(k32[0])];
                        b[3] = p8(30)[p8(31)[p8(32)[b[3]] ^ b3(k32[1])] ^ b3(k32[0])];
                }

        /* Now perform the MDS matrix multiply inline. */
        return  ((M00(b[0]) ^ M01(b[1]) ^ M02(b[2]) ^ M03(b[3]))        ) ^
                        ((M10(b[0]) ^ M11(b[1]) ^ M12(b[2]) ^ M13(b[3])) <<  8) ^
                        ((M20(b[0]) ^ M21(b[1]) ^ M22(b[2]) ^ M23(b[3])) << 16) ^
                        ((M30(b[0]) ^ M31(b[1]) ^ M32(b[2]) ^ M33(b[3])) << 24) ;
        }
#endif  /* CHECK_TABLE */


/*
+*****************************************************************************
*
* Function Name:        RS_MDS_encode
*
* Function:                     Use (12,8) Reed-Solomon code over GF(256) to produce
*                                       a key S-box dword from two key material dwords.
*
* Arguments:            k0      =       1st dword
*                                       k1      =       2nd dword
*
* Return:                       Remainder polynomial generated using RS code
*
* Notes:
*       Since this computation is done only once per reKey per 64 bits of key,
*       the performance impact of this routine is imperceptible. The RS code
*       chosen has "simple" coefficients to allow smartcard/hardware implementation
*       without lookup tables.
*
-*****************************************************************************/
DWORD RS_MDS_Encode(DWORD k0,DWORD k1)
        {
        int i,j;
```

```
        DWORD r;

        for (i=r=0;i<2;i++)
                {
                r ^= (i) ? k0 : k1;                     /* merge in 32 more key bits */
                for (j=0;j<4;j++)                       /* shift one byte at a time */
                        RS_rem(r);
                }
        return r;
        }


/*
+*****************************************************************************
*
* Function Name:        BuildMDS
*
* Function:                     Initialize the MDStab array
*
* Arguments:            None.
*
* Return:                       None.
*
* Notes:
*       Here we precompute all the fixed MDS table.  This only needs to be done
*       one time at initialization, after which the table is "CONST".
*
-*****************************************************************************/
void BuildMDS(void)
        {
        int i;
        DWORD d;
        BYTE m1[2],mX[2],mY[4];

        for (i=0;i<256;i++)
                {
                m1[0]=P8x8[0][i];           /* compute all the matrix elements */
                mX[0]=(BYTE) Mul_X(m1[0]);
                mY[0]=(BYTE) Mul_Y(m1[0]);

                m1[1]=P8x8[1][i];
                mX[1]=(BYTE) Mul_X(m1[1]);
                mY[1]=(BYTE) Mul_Y(m1[1]);

#undef  Mul_1                                   /* change what the pre-processor does with
Mij */
#undef  Mul_X
#undef  Mul_Y
#define Mul_1   m1                              /* It will now access m01[], m5B[], and
mEF[] */
#define Mul_X   mX
#define Mul_Y   mY

#define SetMDS(N)                                       ¥
                b0(d) = M0##N[P_##N##0];        ¥
                b1(d) = M1##N[P_##N##0];        ¥
                b2(d) = M2##N[P_##N##0];        ¥
                b3(d) = M3##N[P_##N##0];        ¥
                MDStab[N][i] = d;
```

```
                SetMDS(0);                              /* fill in the matrix with elements
computed above */
                SetMDS(1);
                SetMDS(2);
                SetMDS(3);
                }
#undef  Mul_1
#undef  Mul_X
#undef  Mul_Y
#define Mul_1   Mx_1                    /* re-enable true multiply */
#define Mul_X   Mx_X
#define Mul_Y   Mx_Y

#if BIG_TAB
        {
        int j,k;
        BYTE *q0,*q1;

        for (i=0;i<4;i++)
                {
                switch (i)
                        {
                        case 0:  q0=p8(01); q1=p8(02);      break;
                        case 1:  q0=p8(11); q1=p8(12);      break;
                        case 2:  q0=p8(21); q1=p8(22);      break;
                        case 3:  q0=p8(31); q1=p8(32);      break;
                        }
                for (j=0;j<256;j++)
                        for (k=0;k<256;k++)
                                bigTab[i][j][k]=q0[q1[k]^j];
                }
        }
#endif

        needToBuildMDS=0;                       /* NEVER modify the table again! */
        }

/*
+*****************************************************************************
*
* Function Name:        ReverseRoundSubkeys
*
* Function:                     Reverse order of round subkeys to switch between
encrypt/decrypt
*
* Arguments:            key             =       ptr to keyInstance to be reversed
*                               newDir  =       new direction value
*
* Return:                       None.
*
* Notes:
*       This optimization allows both blockEncrypt and blockDecrypt to use the same
*       "fallthru" switch statement based on the number of rounds.
*       Note that key->numRounds must be even and >= 2 here.
*
-*****************************************************************************/
void ReverseRoundSubkeys(keyInstance *key,BYTE newDir)
        {
```

```
        DWORD t0,t1;
        register DWORD *r0=key->subKeys+ROUND_SUBKEYS;
        register DWORD *r1=r0 + 2*key->numRounds - 2;

        for (;r0 < r1;r0+=2,r1-=2)
                {
                t0=r0[0];               /* swap the order */
                t1=r0[1];
                r0[0]=r1[0];            /* but keep relative order within pairs */
                r0[1]=r1[1];
                r1[0]=t0;
                r1[1]=t1;
                }

        key->direction=newDir;
        }

/*
+*****************************************************************************
*
* Function Name:        Xor256
*
* Function:                     Copy an 8-bit permutation (256 bytes), xoring with a byte
*
* Arguments:            dst             =       where to put result
*                                       src             =       where to get data (can be
same asa dst)
*                                       b               =       byte to xor
*
* Return:                       None
*
* Notes:
*       BorlandC's optimization is terrible!    When we put the code inline,
*       it generates fairly good code in the *following* segment (not in the Xor256
*       code itself).    If the call is made, the code following the call is awful!
*       The penalty is nearly 50%!    So we take the code size hit for inlining for
*       Borland, while Microsoft happily works with a call.
*
-*****************************************************************************/
#if defined(__BORLANDC__)       /* do it inline */
#define Xor32(dst,src,i) { ((DWORD *)dst)[i] = ((DWORD *)src)[i] ^ tmpX; }
#define  Xor256(dst,src,b)                                      ¥
        {                                                                               ¥
        register DWORD tmpX=0x01010101u * b;¥
        for (i=0;i<64;i+=4)                                                     ¥
                { Xor32(dst,src,i   ); Xor32(dst,src,i+1); Xor32(dst,src,i+2); Xor32(dst,src,i+3); }
        ¥
        }
#else                                                   /* do it as a function call */
void Xor256(void *dst,void *src,BYTE b)
        {
        register DWORD x=b*0x01010101u;         /* replicate byte to all four bytes */
        register DWORD *d=(DWORD *)dst;
        register DWORD *s=(DWORD *)src;
#define X_8(N)    { d[N]=s[N] ^ x; d[N+1]=s[N+1] ^ x; }
#define X_32(N)  { X_8(N); X_8(N+2); X_8(N+4); X_8(N+6); }
        X_32(0 ); X_32( 8); X_32(16); X_32(24);         /* all inline */
        d+=32;   /* keep offsets small! */
        s+=32;
```

```
        X_32(0 ); X_32( 8); X_32(16); X_32(24);       /* all inline */
        }
#endif

/*
+*****************************************************************************
*
* Function Name:        reKey
*
* Function:                     Initialize the Twofish key schedule from key32
*
* Arguments:            key                     =       ptr to keyInstance to be initialized
*
* Return:                       TRUE on success
*
* Notes:
*       Here we precompute all the round subkeys, although that is not actually
*       required.   For example, on a smartcard, the round subkeys can
*       be generated on-the-fly     using f32()
*
-*****************************************************************************/
int reKey(keyInstance *key)
        {
        int             i,j,k64Cnt,keyLen;
        int             subkeyCnt;
        DWORDA=0,B=0,q;
        DWORD
        sKey[MAX_KEY_BITS/64],k32e[MAX_KEY_BITS/64],k32o[MAX_KEY_BITS/64];
        BYTE    L0[256],L1[256]; /* small local 8-bit permutations */

#if VALIDATE_PARMS
  #if ALIGN32
        if (((int)key) & 3)
                return BAD_ALIGN32;
        if ((key->keyLen % 64) || (key->keyLen < MIN_KEY_BITS))
                return BAD_KEY_INSTANCE;
  #endif
#endif

        if (needToBuildMDS)                     /* do this one time only */
                BuildMDS();

#define  F32(res,x,k32)     ¥
        {                                                       ¥
        DWORD t=x;                                      ¥
        switch (k64Cnt & 3)                             ¥
            {                                           ¥
                case 0:   /* same as 4 */               ¥
                {                                       b0(t)     = p8(04)[b0(t)] ^ b0(k32[3]);
        ¥                                               b1(t)     = p8(14)[b1(t)] ^ b1(k32[3]);
        ¥                                               b2(t)     = p8(24)[b2(t)] ^ b2(k32[3]);
        ¥
```

```
                                        b3(t)     = p8(34)[b3(t)] ^ b3(k32[3]);
        ¥
                                    /* fall thru, having pre-processed t */                ¥
                case 3:             b0(t)     = p8(03)[b0(t)] ^ b0(k32[2]);                ¥
                                        b1(t)     = p8(13)[b1(t)] ^ b1(k32[2]);
        ¥
                                        b2(t)     = p8(23)[b2(t)] ^ b2(k32[2]);
        ¥
                                        b3(t)     = p8(33)[b3(t)] ^ b3(k32[2]);
        ¥
                                    /* fall thru, having pre-processed t */                ¥
                case 2:     /* 128-bit keys (optimize for this case) */    ¥
                            res=        MDStab[0][p8(01)[p8(02)[b0(t)] ^ b0(k32[1])] ^ b0(k32[0])] ^
        ¥
                                        MDStab[1][p8(11)[p8(12)[b1(t)]    ^    b1(k32[1])]    ^
b1(k32[0])] ^       ¥
                                        MDStab[2][p8(21)[p8(22)[b2(t)]    ^    b2(k32[1])]    ^
b2(k32[0])] ^       ¥
                                        MDStab[3][p8(31)[p8(32)[b3(t)]    ^    b3(k32[1])]    ^
b3(k32[0])] ;       ¥
                }
                                            ¥
        }


#if !CHECK_TABLE
#if defined(USE_ASM)                        /* only do this if not using assember */
if (!(useAsm & 4))
#endif
#endif
        {
        subkeyCnt = ROUND_SUBKEYS + 2*key->numRounds;
        keyLen=key->keyLen;
        k64Cnt=(keyLen+63)/64;              /* number of 64-bit key words */
        for (i=0,j=k64Cnt-1;i<k64Cnt;i++,j--)
                {                                                          /*    split    into
even/odd key dwords */
                k32e[i]=key->key32[2*i   ];
                k32o[i]=key->key32[2*i+1];
                /* compute S-box keys using (12,8) Reed-Solomon code over GF(256) */
                sKey[j]=key->sboxKeys[j]=RS_MDS_Encode(k32e[i],k32o[i]); /* reverse order */
                }
        }

#ifdef USE_ASM
if (useAsm & 4)
        {
        #if defined(COMPILE_KEY) && defined(USE_ASM)
                key->keySig             = VALID_SIG;                    /* show that we
are initialized */
                key->codeSize   = sizeof(key->compiledCode);    /* set size */
        #endif
        reKey_86(key);
        }
else
#endif
        {
        for (i=q=0;i<subkeyCnt/2;i++,q+=SK_STEP)
                {                                                          /* compute round
```

```
subkeys for PHT */
                F32(A,q          ,k32e);            /* A uses even key dwords */
                F32(B,q+SK_BUMP,k32o);                    /* B uses odd  key dwords */
                B = ROL(B,8);
                key->subKeys[2*i  ] = A+B;         /* combine with a PHT */
                B = A + 2*B;
                key->subKeys[2*i+1] = ROL(B,SK_ROTL);
                }
#if !defined(ZERO_KEY)
        switch (keyLen)  /* case out key length for speed in generating S-boxes */
                {
                case 128:
                #if defined(FULL_KEY) || defined(PART_KEY)
#if BIG_TAB
                        #define  one128(N,J)      sbSet(N,i,J,L0[i+J])
                        #define  sb128(N) {
        ¥
                                BYTE *qq=bigTab[N][b##N(sKey[1])];      ¥
                                Xor256(L0,qq,b##N(sKey[0]));            ¥
                                for (i=0;i<256;i+=2) { one128(N,0); one128(N,1); } }
#else
                        #define  one128(N,J)      sbSet(N,i,J,p8(N##1)[L0[i+J]]^k0)
                        #define  sb128(N) {                                      ¥
                                Xor256(L0,p8(N##2),b##N(sKey[1]));      ¥
                                { register DWORD k0=b##N(sKey[0]);      ¥
                                for (i=0;i<256;i+=2) { one128(N,0); one128(N,1); } } }
#endif
                #elif defined(MIN_KEY)
                        #define  sb128(N) Xor256(_sBox8_(N),p8(N##2),b##N(sKey[1]))
                #endif
                        sb128(0); sb128(1); sb128(2); sb128(3);
                        break;
                case 192:
                #if defined(FULL_KEY) || defined(PART_KEY)
                        #define one192(N,J) sbSet(N,i,J,p8(N##1)[p8(N##2)[L0[i+J]]^k1]^k0)
                        #define  sb192(N) {
        ¥
                                Xor256(L0,p8(N##3),b##N(sKey[2]));      ¥
                                { register DWORD k0=b##N(sKey[0]);      ¥
                                  register DWORD k1=b##N(sKey[1]);      ¥
                                  for (i=0;i<256;i+=2) { one192(N,0); one192(N,1); } } }
                #elif defined(MIN_KEY)
                        #define one192(N,J) sbSet(N,i,J,p8(N##2)[L0[i+J]]^k1)
                        #define  sb192(N) {
        ¥
                                Xor256(L0,p8(N##3),b##N(sKey[2]));      ¥
                                { register DWORD k1=b##N(sKey[1]);      ¥
                                  for (i=0;i<256;i+=2) { one192(N,0); one192(N,1); } } }
                #endif
                        sb192(0); sb192(1); sb192(2); sb192(3);
                        break;
                case 256:
                #if defined(FULL_KEY) || defined(PART_KEY)
                        #define one256(N,J) sbSet(N,i,J,p8(N##1)[p8(N##2)[L0[i+J]]^k1]^k0)
                        #define  sb256(N) {
                                        ¥
                                Xor256(L1,p8(N##4),b##N(sKey[3]));
                        ¥
                                for (i=0;i<256;i+=2) {L0[i  ]=p8(N##3)[L1[i]];
```

```
        ¥

L0[i+1]=p8(N##3)[L1[i+1]]; }        ¥
                                    Xor256(L0,L0,b##N(sKey[2]));
                            ¥
                                    { register DWORD k0=b##N(sKey[0]);
                            ¥
                                      register DWORD k1=b##N(sKey[1]);
                            ¥
                                      for (i=0;i<256;i+=2) { one256(N,0); one256(N,1); } } }
                #elif defined(MIN_KEY)
                        #define one256(N,J) sbSet(N,i,J,p8(N##2)[L0[i+J]]^k1)
                        #define  sb256(N) {
                                        ¥
                                    Xor256(L1,p8(N##4),b##N(sKey[3]));
                            ¥
                                    for (i=0;i<256;i+=2) {L0[i   ]=p8(N##3)[L1[i]];
        ¥

L0[i+1]=p8(N##3)[L1[i+1]]; }        ¥
                                    Xor256(L0,L0,b##N(sKey[2]));
                            ¥
                                    { register DWORD k1=b##N(sKey[1]);
                            ¥
                                      for (i=0;i<256;i+=2) { one256(N,0); one256(N,1); } } }
                #endif
                        sb256(0); sb256(1);         sb256(2); sb256(3);
                        break;
                }
#endif
        }

#if CHECK_TABLE                                                 /*  sanity  check  vs.
pedagogical code*/
        {
        GetSboxKey;
        for (i=0;i<subkeyCnt/2;i++)
                {
                A = f32(i*SK_STEP           ,k32e,keyLen);  /* A uses even key dwords */
                B = f32(i*SK_STEP+SK_BUMP,k32o,keyLen);         /* B uses odd   key dwords
*/
                B = ROL(B,8);
                assert(key->subKeys[2*i   ] == A+   B);
                assert(key->subKeys[2*i+1] == ROL(A+2*B,SK_ROTL));
                }
  #if !defined(ZERO_KEY)                                /* any S-boxes to check? */
        for (i=q=0;i<256;i++,q+=0x01010101)
                assert(f32(q,key->sboxKeys,keyLen) == Fe32_(q,0));
  #endif
        }
#endif /* CHECK_TABLE */

        DebugDumpKey(key);

        if (key->direction == DIR_ENCRYPT)
                ReverseRoundSubkeys(key,DIR_ENCRYPT);           /*  reverse  the  round
subkey order */

        return TRUE;
```

```
        }
/*
+*****************************************************************************
*
* Function Name:        makeKey
*
* Function:                     Initialize the Twofish key schedule
*
* Arguments:            key                     =       ptr to keyInstance to be initialized
*                                       direction       =       DIR_ENCRYPT or
DIR_DECRYPT
*                                       keyLen          =       # bits of key text at
*keyMaterial
*                                       keyMaterial     =       ptr to hex ASCII chars
representing key bits
*
* Return:                       TRUE on success
*                                       else error code (e.g., BAD_KEY_DIR)
*
* Notes: This parses the key bits from keyMaterial.   Zeroes out unused key bits
*
-*****************************************************************************/
int makeKey(keyInstance *key, BYTE direction, int keyLen,CONST char *keyMaterial)
        {
#if VALIDATE_PARMS                              /* first, sanity check on parameters */
        if (key == NULL)
                return BAD_KEY_INSTANCE;/* must have a keyInstance to initialize */
        if ((direction != DIR_ENCRYPT) && (direction != DIR_DECRYPT))
                return BAD_KEY_DIR;             /* must have valid direction */
        if ((keyLen > MAX_KEY_BITS) || (keyLen < 8) || (keyLen & 0x3F))
                return BAD_KEY_MAT;             /* length must be valid */
        key->keySig = VALID_SIG;        /* show that we are initialized */
  #if ALIGN32
        if ((((int)key) & 3) || (((int)key->key32) & 3))
                return BAD_ALIGN32;
  #endif
#endif

        key->direction  = direction;/* set our cipher direction */
        key->keyLen             = (keyLen+63) & ~63;            /* round up to multiple of
64 */
        key->numRounds          = numRounds[(keyLen-1)/64];
        memset(key->key32,0,sizeof(key->key32));  /* zero unused bits */
        key->keyMaterial[MAX_KEY_SIZE]=0;       /* terminate ASCII string */

        if ((keyMaterial == NULL) || (keyMaterial[0]==0))
                return TRUE;                    /* allow a "dummy" call */

        if (ParseHexDword(keyLen,keyMaterial,key->key32,key->keyMaterial))
                return BAD_KEY_MAT;

        return reKey(key);                      /* generate round subkeys */
        }


/*
+*****************************************************************************
*
* Function Name:        cipherInit
```

```
*
* Function:                     Initialize the Twofish cipher in a given mode
*
* Arguments:        cipher      =       ptr to cipherInstance to be initialized
*                               mode            =       MODE_ECB,
MODE_CBC, or MODE_CFB1
*                               IV              =       ptr to hex ASCII
test representing IV bytes
*
* Return:                       TRUE on success
*                               else error code (e.g., BAD_CIPHER_MODE)
*
-*****************************************************************************/
int cipherInit(cipherInstance *cipher, BYTE mode,CONST char *IV)
        {
        int i;
#if VALIDATE_PARMS                              /* first, sanity check on parameters */
        if (cipher == NULL)
                return BAD_PARAMS;              /* must have a cipherInstance to initialize */
        if ((mode != MODE_ECB) && (mode != MODE_CBC) && (mode != MODE_CFB1))
                return BAD_CIPHER_MODE;         /* must have valid cipher mode */
        cipher->cipherSig       =       VALID_SIG;
  #if ALIGN32
        if ((((int)cipher) & 3) | | (((int)cipher->IV) & 3) | | (((int)cipher->iv32) & 3))
                return BAD_ALIGN32;
  #endif
#endif

        if ((mode != MODE_ECB) && (IV)) /* parse the IV */
                {
                if (ParseHexDword(BLOCK_SIZE,IV,cipher->iv32,NULL))
                        return BAD_IV_MAT;
                for (i=0;i<BLOCK_SIZE/32;i++)   /* make byte-oriented copy for CFB1 */
                        ((DWORD *)cipher->IV)[i] = Bswap(cipher->iv32[i]);
                }

        cipher->mode            =       mode;

        return TRUE;
        }

/*
+*****************************************************************************
*
* Function Name:        blockEncrypt
*
* Function:                     Encrypt block(s) of data using Twofish
*
* Arguments:        cipher      =       ptr to already initialized cipherInstance
*                               key                     =       ptr to already
initialized keyInstance
*                               input           =       ptr to data blocks to be
encrypted
*                               inputLen        =       # bits to encrypt (multiple
of blockSize)
*                               outBuffer       =       ptr to where to put
encrypted blocks
*
* Return:                       # bits ciphered (>= 0)
```

```
*                                                   else    error    code    (e.g.,    BAD_CIPHER_STATE,
BAD_KEY_MATERIAL)
*
* Notes: The only supported block size for ECB/CBC modes is BLOCK_SIZE bits.
*                   If inputLen is not a multiple of BLOCK_SIZE bits in those modes,
*                   an error BAD_INPUT_LEN is returned.   In CFB1 mode, all block
*                   sizes can be supported.
*
-*************************************************************************/
int blockEncrypt(cipherInstance *cipher, keyInstance *key,CONST BYTE *input,
                                    int inputLen, BYTE *outBuffer)
        {
        int     i,n;                                                    /* loop counters */
        DWORD x[BLOCK_SIZE/32];                         /* block being encrypted */
        DWORD t0,t1;                                            /* temp variables */
        int             rounds=key->numRounds;          /* number of rounds */
        BYTE    bit,bit0,ctBit,carry;                   /* temps for CFB */

        /* make local copies of things for faster access */
        int             mode = cipher->mode;
        DWORD sk[TOTAL_SUBKEYS];
        DWORD IV[BLOCK_SIZE/32];

        GetSboxKey;

#if VALIDATE_PARMS
        if ((cipher == NULL) || (cipher->cipherSig != VALID_SIG))
                return BAD_CIPHER_STATE;
        if ((key == NULL) || (key->keySig != VALID_SIG))
                return BAD_KEY_INSTANCE;
        if ((rounds < 2) || (rounds > MAX_ROUNDS) || (rounds&1))
                return BAD_KEY_INSTANCE;
        if ((mode != MODE_CFB1) && (inputLen % BLOCK_SIZE))
                return BAD_INPUT_LEN;
  #if ALIGN32
        if ( (((int)cipher) & 3) || (((int)key          ) & 3) ||
                 (((int)input ) & 3) || (((int)outBuffer) & 3))
                return BAD_ALIGN32;
  #endif
#endif

        if (mode == MODE_CFB1)
                {       /* use recursion here to handle CFB, one block at a time */
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,cipher->IV,BLOCK_SIZE,(BYTE *)x);
                        bit0    = 0x80 >> (n & 7);/* which bit position in byte */
                        ctBit = (input[n/8] & bit0) ^ ((((BYTE *) x)[0] & 0x80) >> (n&7));
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) | ctBit;
                        carry = ctBit >> (7 - (n&7));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;       /* restore mode for next time */
```

```
                return inputLen;
                }

        /* here for ECB, CBC modes */
        if (key->direction != DIR_ENCRYPT)
                ReverseRoundSubkeys(key,DIR_ENCRYPT);          /* reverse the round
subkey order */

#ifdef USE_ASM
        if ((useAsm & 1) && (inputLen))
  #ifdef COMPILE_KEY
                if (key->keySig == VALID_SIG)
                        return                          ((CipherProc                    *)(key-
>encryptFuncPtr))(cipher,key,input,inputLen,outBuffer);
  #else
                return (*blockEncrypt_86)(cipher,key,input,inputLen,outBuffer);
  #endif
#endif
        /* make local copy of subkeys for speed */
        memcpy(sk,key->subKeys,sizeof(DWORD)*(ROUND_SUBKEYS+2*rounds));
        if (mode == MODE_CBC)
                BlockCopy(IV,cipher->iv32)
        else
                IV[0]=IV[1]=IV[2]=IV[3]=0;

        for
(n=0;n<inputLen;n+=BLOCK_SIZE,input+=BLOCK_SIZE/8,outBuffer+=BLOCK_SIZE/8)
                {
#ifdef DEBUG
                DebugDump(input,"¥n",-1,0,0,0,1);
                if (cipher->mode == MODE_CBC)
                        DebugDump(cipher->iv32,"",IV_ROUND,0,0,0,0);
#endif
#define  LoadBlockE(N)     x[N]=Bswap(((DWORD  *)input)[N])  ^  sk[INPUT_WHITEN+N]  ^
IV[N]
                LoadBlockE(0);  LoadBlockE(1);  LoadBlockE(2);  LoadBlockE(3);
                DebugDump(x,"",0,0,0,0,0);
#define  EncryptRound(K,R,id)      ¥
                        t0              = Fe32##id(x[K    ],0);
        ¥
                        t1              = Fe32##id(x[K^1],3);
        ¥
                        x[K^3] = ROL(x[K^3],1);
                ¥
                        x[K^2]^= t0 +     t1 + sk[ROUND_SUBKEYS+2*(R)    ];        ¥
                        x[K^3]^= t0 + 2*t1 + sk[ROUND_SUBKEYS+2*(R)+1];        ¥
                        x[K^2] = ROR(x[K^2],1);
                ¥
                        DebugDump(x,"",rounds-(R),0,0,1,0);
#define         Encrypt2(R,id)    { EncryptRound(0,R+1,id); EncryptRound(2,R,id); }

#if defined(ZERO_KEY)
                switch (key->keyLen)
                        {
                        case 128:
                                for (i=rounds-2;i>=0;i-=2)
                                        Encrypt2(i,_128);
                                break;
                        case 192:
```

```
                                        for (i=rounds-2;i>=0;i-=2)
                                                Encrypt2(i,_192);
                                        break;
                                case 256:
                                        for (i=rounds-2;i>=0;i-=2)
                                                Encrypt2(i,_256);
                                        break;
                                }
#else
                        Encrypt2(14,_);
                        Encrypt2(12,_);
                        Encrypt2(10,_);
                        Encrypt2( 8,_);
                        Encrypt2( 6,_);
                        Encrypt2( 4,_);
                        Encrypt2( 2,_);
                        Encrypt2( 0,_);
#endif

                        /* need to do (or undo, depending on your point of view) final swap */
#if LittleEndian
#define StoreBlockE(N)  ((DWORD *)outBuffer)[N]=x[N^2] ^ sk[OUTPUT_WHITEN+N]
#else
#define StoreBlockE(N)  {       t0=x[N^2]       ^       sk[OUTPUT_WHITEN+N];       ((DWORD
*)outBuffer)[N]=Bswap(t0); }
#endif
                        StoreBlockE(0);  StoreBlockE(1);  StoreBlockE(2);  StoreBlockE(3);
                        if (mode == MODE_CBC)
                                {
                                IV[0]=Bswap(((DWORD *)outBuffer)[0]);
                                IV[1]=Bswap(((DWORD *)outBuffer)[1]);
                                IV[2]=Bswap(((DWORD *)outBuffer)[2]);
                                IV[3]=Bswap(((DWORD *)outBuffer)[3]);
                                }
#ifdef DEBUG
                        DebugDump(outBuffer,"",rounds+1,0,0,0,1);
                        if (cipher->mode == MODE_CBC)
                                DebugDump(cipher->iv32,"",IV_ROUND,0,0,0,0);
#endif
                        }

                if (mode == MODE_CBC)
                        BlockCopy(cipher->iv32,IV);

                return inputLen;
                }

/*
+*****************************************************************************
*
* Function Name:        blockDecrypt
*
* Function:                     Decrypt block(s) of data using Twofish
*
* Arguments:            cipher          =       ptr to already initialized cipherInstance
*                                       key                     =       ptr     to      already
initialized keyInstance
*                                       input           =       ptr     to      data    blocks  to      be
decrypted
```

```
*                                       inputLen        =       # bits to encrypt (multiple
of blockSize)
*                                       outBuffer       =       ptr to where to put
decrypted blocks
*
* Return:                       # bits ciphered (>= 0)
*                                       else error code (e.g., BAD_CIPHER_STATE,
BAD_KEY_MATERIAL)
*
* Notes: The only supported block size for ECB/CBC modes is BLOCK_SIZE bits.
*               If inputLen is not a multiple of BLOCK_SIZE bits in those modes,
*               an error BAD_INPUT_LEN is returned.  In CFB1 mode, all block
*               sizes can be supported.
*
-*****************************************************************************/
int blockDecrypt(cipherInstance *cipher, keyInstance *key,CONST BYTE *input,
                                int inputLen, BYTE *outBuffer)
        {
        int     i,n;                                            /* loop counters */
        DWORD x[BLOCK_SIZE/32];                         /* block being encrypted */
        DWORD t0,t1;                                    /* temp variables */
        int         rounds=key->numRounds;      /* number of rounds */
        BYTE    bit,bit0,ctBit,carry;           /* temps for CFB */

        /* make local copies of things for faster access */
        int         mode = cipher->mode;
        DWORD sk[TOTAL_SUBKEYS];
        DWORD IV[BLOCK_SIZE/32];

        GetSboxKey;

#if VALIDATE_PARMS
        if ((cipher == NULL) || (cipher->cipherSig != VALID_SIG))
                return BAD_CIPHER_STATE;
        if ((key == NULL) || (key->keySig != VALID_SIG))
                return BAD_KEY_INSTANCE;
        if ((rounds < 2) || (rounds > MAX_ROUNDS) || (rounds&1))
                return BAD_KEY_INSTANCE;
        if ((cipher->mode != MODE_CFB1) && (inputLen % BLOCK_SIZE))
                return BAD_INPUT_LEN;
  #if ALIGN32
        if ( (((int)cipher) & 3) || (((int)key      ) & 3) ||
                 (((int)input)  & 3) || (((int)outBuffer) & 3))
                return BAD_ALIGN32;
  #endif
#endif

        if (cipher->mode == MODE_CFB1)
                {       /* use blockEncrypt here to handle CFB, one block at a time */
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,cipher->IV,BLOCK_SIZE,(BYTE *)x);
                        bit0  = 0x80 >> (n & 7);
                        ctBit = input[n/8] & bit0;
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) |
                                                (ctBit ^ ((((BYTE *) x)[0] & 0x80)
>> (n&7)));
                        carry = ctBit >> (7 - (n&7));
```

```
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;       /* restore mode for next time */
                return inputLen;
                }

        /* here for ECB, CBC modes */
        if (key->direction != DIR_DECRYPT)
                ReverseRoundSubkeys(key,DIR_DECRYPT);           /* reverse the round
subkey order */
#ifdef USE_ASM
        if ((useAsm & 2) && (inputLen))
  #ifdef COMPILE_KEY
                if (key->keySig == VALID_SIG)
                        return                          ((CipherProc                    *)(key-
>decryptFuncPtr))(cipher,key,input,inputLen,outBuffer);
  #else
                return (*blockDecrypt_86)(cipher,key,input,inputLen,outBuffer);
  #endif
#endif
        /* make local copy of subkeys for speed */
        memcpy(sk,key->subKeys,sizeof(DWORD)*(ROUND_SUBKEYS+2*rounds));
        if (mode == MODE_CBC)
                BlockCopy(IV,cipher->iv32)
        else
                IV[0]=IV[1]=IV[2]=IV[3]=0;

        for
(n=0;n<inputLen;n+=BLOCK_SIZE,input+=BLOCK_SIZE/8,outBuffer+=BLOCK_SIZE/8)
                {
                DebugDump(input,"¥n",rounds+1,0,0,0,1);
#define LoadBlockD(N) x[N^2]=Bswap(((DWORD *)input)[N]) ^ sk[OUTPUT_WHITEN+N]
                LoadBlockD(0);  LoadBlockD(1);  LoadBlockD(2);  LoadBlockD(3);

#define  DecryptRound(K,R,id)
        ¥
                        t0              = Fe32##id(x[K    ],0);
        ¥
                        t1              = Fe32##id(x[K^1],3);
        ¥
                        DebugDump(x,"",(R)+1,0,0,1,0);
        ¥
                        x[K^2] = ROL (x[K^2],1);
        ¥
                        x[K^2]^= t0 +     t1 + sk[ROUND_SUBKEYS+2*(R)    ];       ¥
                        x[K^3]^= t0 + 2*t1 + sk[ROUND_SUBKEYS+2*(R)+1];       ¥
                        x[K^3] = ROR (x[K^3],1);
        ¥

#define         Decrypt2(R,id)    { DecryptRound(2,R+1,id); DecryptRound(0,R,id); }

#if defined(ZERO_KEY)
                switch (key->keyLen)
                        {
```

```
                        case 128:
                                for (i=rounds-2;i>=0;i-=2)
                                        Decrypt2(i,_128);
                                break;
                        case 192:
                                for (i=rounds-2;i>=0;i-=2)
                                        Decrypt2(i,_192);
                                break;
                        case 256:
                                for (i=rounds-2;i>=0;i-=2)
                                        Decrypt2(i,_256);
                                break;
                        }
#else
                {
                Decrypt2(14,_);
                Decrypt2(12,_);
                Decrypt2(10,_);
                Decrypt2( 8,_);
                Decrypt2( 6,_);
                Decrypt2( 4,_);
                Decrypt2( 2,_);
                Decrypt2( 0,_);
                }
#endif
                DebugDump(x,"",0,0,0,0,0);
                if (cipher->mode == MODE_ECB)
                        {
#if LittleEndian
#define  StoreBlockD(N)  ((DWORD *)outBuffer)[N] = x[N] ^ sk[INPUT_WHITEN+N]
#else
#define  StoreBlockD(N)  {  t0=x[N]^sk[INPUT_WHITEN+N];  ((DWORD  *)outBuffer)[N]  =
Bswap(t0); }
#endif
                        StoreBlockD(0);  StoreBlockD(1);  StoreBlockD(2);  StoreBlockD(3);
#undef   StoreBlockD
                        DebugDump(outBuffer,"",-1,0,0,0,1);
                        continue;
                        }
                else
                        {
#define  StoreBlockD(N)  x[N]    ^= sk[INPUT_WHITEN+N] ^ IV[N];          ¥
                                                IV[N]     =  Bswap(((DWORD  *)input)[N]);
        ¥
                                                ((DWORD *)outBuffer)[N] = Bswap(x[N]);
                        StoreBlockD(0);  StoreBlockD(1);  StoreBlockD(2);  StoreBlockD(3);
#undef   StoreBlockD
                        DebugDump(outBuffer,"",-1,0,0,0,1);
                        }
                }
        if (mode == MODE_CBC)  /* restore iv32 to cipher */
                BlockCopy(cipher->iv32,IV)

        return inputLen;
        }

#ifdef GetCodeSize
DWORD TwofishCodeSize(void)
        {
```

```
	DWORD x= Here(0);
#ifdef USE_ASM
	if (useAsm & 3)
		return TwofishAsmCodeSize();
#endif
	return x - TwofishCodeStart();
	};
#endif


////////////////////////////////////////////////////////////////////

/* aes.h */

/* ---------- See examples at end of this file for typical usage -------- */

/* AES Cipher header file for ANSI C Submissions
	Lawrence E. Bassham III
	Computer Security Division
	National Institute of Standards and Technology

	This sample is to assist implementers developing to the
Cryptographic API Profile for AES Candidate Algorithm Submissions.
Please consult this document as a cross-reference.

	ANY CHANGES, WHERE APPROPRIATE, TO INFORMATION PROVIDED IN THIS FILE
MUST BE DOCUMENTED. CHANGES ARE ONLY APPROPRIATE WHERE SPECIFIED WITH
THE STRING "CHANGE POSSIBLE". FUNCTION CALLS AND THEIR PARAMETERS
CANNOT BE CHANGED. STRUCTURES CAN BE ALTERED TO ALLOW IMPLEMENTERS TO
INCLUDE IMPLEMENTATION SPECIFIC INFORMATION.
*/

/* Includes:
	Standard include files
*/

#include<stdio.h>
#include"platform.h"			/* platform-specific defines */

/*	Defines:
		Add any additional defines you need
*/

#define	DIR_ENCRYPT	0		/* Are we encrpyting? */
#define	DIR_DECRYPT	1		/* Are we decrpyting? */
#define	MODE_ECB		1		/* Are we ciphering in ECB mode? */
#define	MODE_CBC		2		/* Are we ciphering in CBC mode? */
#define	MODE_CFB1		3		/* Are we ciphering in 1-bit CFB mode? */

#define	TRUE			1
#define	FALSE			0

#define	BAD_KEY_DIR		-1	/* Key direction is invalid (unknown value) */
#define	BAD_KEY_MAT			-2	/* Key material not of correct length */
#define	BAD_KEY_INSTANCE	-3	/* Key passed is not valid */
#define	BAD_CIPHER_MODE	-4	/* Params struct passed to cipherInit invalid */
```

```
#define  BAD_CIPHER_STATE     -5        /* Cipher in wrong state (e.g., not initialized) */

/* CHANGE POSSIBLE: inclusion of algorithm specific defines */
/* TWOFISH specific definitions */
#define             MAX_KEY_SIZE            64      /* # of ASCII chars needed to represent a key
*/
#define             MAX_IV_SIZE                     16      /* # of bytes needed to represent an
IV */
#define             BAD_INPUT_LEN                   -6      /* inputLen not a multiple of block
size */
#define             BAD_PARAMS                      -7      /* invalid parameters */
#define             BAD_IV_MAT                      -8      /* invalid IV text */
#define             BAD_ENDIAN                      -9      /* incorrect endianness define */
#define             BAD_ALIGN32                     -10     /* incorrect 32-bit alignment */

#define             BLOCK_SIZE                      128     /* number of bits per block */
#define             MAX_ROUNDS                      16      /* max # rounds (for allocating
subkey array) */
#define             ROUNDS_128                      16      /* default number of rounds for
128-bit keys*/
#define             ROUNDS_192                      16      /* default number of rounds for
192-bit keys*/
#define             ROUNDS_256                      16      /* default number of rounds for
256-bit keys*/
#define             MAX_KEY_BITS            256     /* max number of bits of key */
#define             MIN_KEY_BITS            128     /* min number of bits of key (zero pad) */
#define             VALID_SIG       0x48534946      /* initialization signature ('FISH') */
#define             MCT_OUTER                       400     /* MCT outer loop */
#define             MCT_INNER                 10000 /* MCT inner loop */
#define             REENTRANT                       1       /* nonzero forces reentrant code
(slightly slower) */

#define             INPUT_WHITEN                    0       /* subkey array indices */
#define             OUTPUT_WHITEN                   ( INPUT_WHITEN + BLOCK_SIZE/32)
#define             ROUND_SUBKEYS                   (OUTPUT_WHITEN + BLOCK_SIZE/32)
        /* use 2 * (# rounds) */
#define             TOTAL_SUBKEYS                   (ROUND_SUBKEYS + 2*MAX_ROUNDS)

/* Typedefs:
        Typedef'ed data storage elements. Add any algorithm specific
        parameters at the bottom of the structs as appropriate.
*/

typedef unsigned char BYTE;
typedef  unsigned long DWORD;           /* 32-bit unsigned quantity */
typedef DWORD fullSbox[4][256];

/* The structure for key information */
typedef struct
        {
        BYTE direction;                                 /* Key used for encrypting or
decrypting? */
#if ALIGN32
        BYTE dummyAlign[3];                             /* keep 32-bit alignment */
#endif
        int   keyLen;                                   /* Length of the key */
        char keyMaterial[MAX_KEY_SIZE+4];/* Raw key data in ASCII */

        /* Twofish-specific parameters: */
```

```
        DWORD keySig;                                   /* set to VALID_SIG by makeKey()
*/
        int         numRounds;                          /* number of rounds in cipher */
        DWORD key32[MAX_KEY_BITS/32];       /* actual key bits, in dwords */
        DWORD sboxKeys[MAX_KEY_BITS/64];/* key bits used for S-boxes */
        DWORD subKeys[TOTAL_SUBKEYS];       /* round subkeys, input/output whitening
bits */
#if REENTRANT
        fullSbox sBox8x32;                              /* fully expanded S-box */
  #if defined(COMPILE_KEY) && defined(USE_ASM)
#undef  VALID_SIG
#define VALID_SIG       0x504D4F43              /* 'COMP':     C is compiled with -
DCOMPILE_KEY */
        DWORD cSig1;                                    /* set after first "compile" (zero at
"init") */
        void *encryptFuncPtr;                   /* ptr to asm encrypt function */
        void *decryptFuncPtr;                   /* ptr to asm decrypt function */
        DWORD codeSize;                                 /* size of compiledCode */
        DWORD cSig2;                                    /* set after first "compile" */
        BYTE    compiledCode[5000];             /* make room for the code itself */
  #endif
#endif
        } keyInstance;

/* The structure for cipher information */
typedef struct
        {
        BYTE    mode;                                   /*          MODE_ECB,
MODE_CBC, or MODE_CFB1 */
#if ALIGN32
        BYTE dummyAlign[3];                             /* keep 32-bit alignment */
#endif
        BYTE    IV[MAX_IV_SIZE];                        /* CFB1 iv bytes   (CBC uses iv32)
*/

        /* Twofish-specific parameters: */
        DWORD cipherSig;                                /* set to VALID_SIG by cipherInit()
*/
        DWORD iv32[BLOCK_SIZE/32];          /* CBC IV bytes arranged as dwords */
        } cipherInstance;

/* Function protoypes */
int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial);

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                                    int inputLen, BYTE *outBuffer);

int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                                    int inputLen, BYTE *outBuffer);

int     reKey(keyInstance *key); /* do key schedule using modified key.keyDwords */

/* API to check table usage, for use in ECB_TBL KAT */
#define         TAB_DISABLE                     0
#define         TAB_ENABLE                      1
#define         TAB_RESET                       2
#define         TAB_QUERY                       3
```

```
#define     TAB_MIN_QUERY           50
int TableOp(int op);


#define     CONST                   /* helpful C++ syntax sugar, NOP for ANSI
C */

#if BLOCK_SIZE == 128               /* optimize block copies */
#define     Copy1(d,s,N)    ((DWORD *)(d))[N] = ((DWORD *)(s))[N]
#define     BlockCopy(d,s)  { Copy1(d,s,0);Copy1(d,s,1);Copy1(d,s,2);Copy1(d,s,3); }
#else
#define     BlockCopy(d,s)  { memcpy(d,s,BLOCK_SIZE/8); }
#endif


#ifdef TEST_2FISH
/*                                      ----- EXAMPLES -----

Unfortunately, the AES API is somewhat clumsy, and it is not entirely
obvious how to use the above functions.  In particular, note that
makeKey() takes an ASCII hex nibble key string (e.g., 32 characters
for a 128-bit key), which is rarely the way that keys are internally
represented.  The reKey() function uses instead the keyInstance.key32
array of key bits and is the preferred method.  In fact, makeKey()
initializes some internal keyInstance state, then parse the ASCII
string into the binary key32, and calls reKey().  To initialize the
keyInstance state, use a 'dummy' call to makeKey(); i.e., set the
keyMaterial parameter to NULL.  Then use reKey() for all key changes.
Similarly, cipherInit takes an IV string in ASCII hex, so a dummy setup
call with a null IV string will skip the ASCII parse.

Note that CFB mode is not well tested nor defined by AES, so using the
Twofish MODE_CFB it not recommended.  If you wish to implement a CFB mode,
build it external to the Twofish code, using the Twofish functions only
in ECB mode.

Below is a sample piece of code showing how the code is typically used
to set up a key, encrypt, and decrypt.  Error checking is somewhat limited
in this example.  Pseudorandom bytes are used for all key and text.

If you compile TWOFISH2.C or TWOFISH.C as a DOS (or Windows Console) app
with this code enabled, the test will be run.  For example, using
Borland C, you would compile using:
  BCC32 -DTEST_2FISH twofish2.c
to run the test on the optimized code, or
  BCC32 -DTEST_2FISH twofish.c
to run the test on the pedagogical code.

*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#define MAX_BLK_CNT         4           /* max # blocks per call in TestTwofish */
int TestTwofish(int mode,int keySize) /* keySize must be 128, 192, or 256 */
        {                                                   /* return 0 iff test passes
*/
```

```
        keyInstance      ki;                            /* key information, including tables */
        cipherInstance ci;                              /* keeps mode (ECB, CBC) and IV */
        BYTE    plainText[MAX_BLK_CNT*(BLOCK_SIZE/8)];
        BYTE cipherText[MAX_BLK_CNT*(BLOCK_SIZE/8)];
        BYTE decryptOut[MAX_BLK_CNT*(BLOCK_SIZE/8)];
        BYTE iv[BLOCK_SIZE/8];
        int    i,byteCnt;

        if (makeKey(&ki,DIR_ENCRYPT,keySize,NULL) != TRUE)
                return 1;                               /* 'dummy' setup for a 128-bit key
*/
        if (cipherInit(&ci,mode,NULL) != TRUE)
                return 1;                               /* 'dummy' setup for cipher */

        for (i=0;i<keySize/32;i++)  /* select key bits */
                ki.key32[i]=0x10003 * rand();
        reKey(&ki);                                     /* run the key schedule */

        if (mode != MODE_ECB)               /* set up random iv (if needed)*/
                {
                for (i=0;i<sizeof(iv);i++)
                        iv[i]=(BYTE) rand();
                memcpy(ci.iv32,iv,sizeof(ci.iv32));   /* copy the IV to ci */
                }

        /* select number of bytes to encrypt (multiple of block) */
        /* e.g., byteCnt = 16, 32, 48, 64 */
        byteCnt = (BLOCK_SIZE/8) * (1 + (rand() % MAX_BLK_CNT));

        for (i=0;i<byteCnt;i++)                 /* generate test data */
                plainText[i]=(BYTE) rand();

        /* encrypt the bytes */
        if (blockEncrypt(&ci,&ki, plainText,byteCnt*8,cipherText) != byteCnt*8)
                return 1;

        /* decrypt the bytes */
        if (mode != MODE_ECB)               /* first re-init the IV (if needed) */
                memcpy(ci.iv32,iv,sizeof(ci.iv32));

        if (blockDecrypt(&ci,&ki,cipherText,byteCnt*8,decryptOut) != byteCnt*8)
                return 1;

        /* make sure the decrypt output matches original plaintext */
        if (memcmp(plainText,decryptOut,byteCnt))
                return 1;

        return 0;                                       /* tests passed! */
        }

void main(void)
        {
        int testCnt,keySize;

        srand((unsigned) time(NULL));      /* randomize */

        for (keySize=128;keySize<=256;keySize+=64)
                for (testCnt=0;testCnt<10;testCnt++)
                        {
```

```
                        if (TestTwofish(MODE_ECB,keySize))
                                { printf("ECB Failure at keySize=%d",keySize); return; }
                        if (TestTwofish(MODE_CBC,keySize))
                                { printf("CBC Failure at keySize=%d",keySize); return; }
                        }
        printf("Tests passed");
        }
#endif /* TEST_2FISH */


////////////////////////////////////////////////////////////////////

#ifdef DEBUG    /* keep these macros common so they are same for both versions */
CONST int debugCompile=         1;
extern  int     debug;
extern  void DebugIO(CONST char *s);    /* display the debug output */

#define DebugDump(x,s,R,XOR,doRot,showT,needBswap)    ¥
        { if (debug) _Dump(x,s,R,XOR,doRot,showT,needBswap,t0,t1); }
#define  DebugDumpKey(key) { if (debug) _DumpKey(key); }
#define  IV_ROUND       -100

void _Dump(CONST void *p,CONST char *s,int R,int XOR,int doRot,int showT,int needBswap,
                DWORD t0,DWORD t1)
        {
        char line[512];    /* build output here */
        int   i,n;
        DWORD q[4];

        if (R == IV_ROUND)
                sprintf(line,"%sIV:     ",s);
        else
                sprintf(line,"%sR[%2d]: ",s,R);
        for (n=0;line[n];n++) ;

        for (i=0;i<4;i++)
                {
                q[i]=((CONST DWORD *)p)[i^(XOR)];
                if (needBswap) q[i]=Bswap(q[i]);
                }

        sprintf(line+n,"x= %08lX  %08lX  %08lX  %08lX.",
                        ROR(q[0],doRot*(R  )/2),
                        ROL(q[1],doRot*(R  )/2),
                        ROR(q[2],doRot*(R+1)/2),
                        ROL(q[3],doRot*(R+1)/2));
        for (;line[n];n++) ;

        if (showT)
                sprintf(line+n,"     t0=%08lX. t1=%08lX.",t0,t1);
        for (;line[n];n++) ;

        sprintf(line+n,"¥n");
        DebugIO(line);
        }

void _DumpKey(CONST keyInstance *key)
        {
        char    line[512];
```

```
        int             i;
        int             k64Cnt=(key->keyLen+63)/64;     /* round up to next multiple of 64
bits */
        int             subkeyCnt = ROUND_SUBKEYS + 2*key->numRounds;

        sprintf(line,";¥n;makeKey:  Input key               -->  S-box key      [%s]¥n",
                (key->direction == DIR_ENCRYPT) ? "Encrypt" : "Decrypt");
        DebugIO(line);
        for (i=0;i<k64Cnt;i++)      /* display in RS format */
                {
                sprintf(line,";%12s %08lX %08lX   -->   %08lX¥n","",
                        key->key32[2*i+1],key->key32[2*i],key->sboxKeys[k64Cnt-1-i]);
                DebugIO(line);
                }
        sprintf(line,";%11sSubkeys¥n","");
        DebugIO(line);
        for (i=0;i<subkeyCnt/2;i++)
                {
                sprintf(line,";%12s          %08lX          %08lX%s¥n","",key->subKeys[2*i],key-
>subKeys[2*i+1],
                        (2*i ==   INPUT_WHITEN) ? "    Input whiten" :
                        (2*i == OUTPUT_WHITEN) ? "   Output whiten" :
                    (2*i == ROUND_SUBKEYS) ? "   Round subkeys" : "");
                DebugIO(line);
                }
        DebugIO(";¥n");
        }
#else
CONST int debugCompile=         0;
#define DebugDump(x,s,R,XOR,doRot,showT,needBswap)
#define  DebugDumpKey(key)
#endif


////////////////////////////////////////////////////

/*****************************************************************************
        PLATFORM.H    -- Platform-specific defines for TWOFISH code

        Submitters:
                Bruce Schneier, Counterpane Systems
                Doug Whiting,   Hi/fn
                John Kelsey,    Counterpane Systems
                Chris Hall,             Counterpane Systems
                David Wagner,   UC Berkeley

        Code Author:            Doug Whiting,    Hi/fn

        Version  1.00           April 1998

        Copyright 1998, Hi/fn and Counterpane Systems.  All rights reserved.

        Notes:
                *       Tab size is set to 4 characters in this file

*****************************************************************************/

/* use intrinsic rotate if possible */
#define  ROL(x,n) (((x) << ((n) & 0x1F)) | ((x) >> (32-((n) & 0x1F))))
```

```
#define  ROR(x,n) (((x) >> ((n) & 0x1F)) | ((x) << (32-((n) & 0x1F))))

#if (0) && defined(__BORLANDC__) && (__BORLANDC__ >= 0x462)
#error "!!!This does not work for some reason!!!"
#include<stdlib.h>                                  /* get prototype for _lrotl() , _lrotr()
*/
#pragma inline __lrotl__
#pragma inline __lrotr__
#undef   ROL                                                    /*   get   rid   of
inefficient definitions */
#undef   ROR
#define  ROL(x,n)          __lrotl__(x,n)          /* use compiler intrinsic rotations */
#define  ROR(x,n)          __lrotr__(x,n)
#endif

#ifdef _MSC_VER
#include<stdlib.h>                                  /*   get   prototypes   for   rotation
functions */
#undef   ROL
#undef   ROR
#pragma intrinsic(_lrotl,_lrotr)            /* use intrinsic compiler rotations */
#define  ROL(x,n)          _lrotl(x,n)
#define  ROR(x,n)          _lrotr(x,n)
#endif

#ifndef _M_IX86
#ifdef    __BORLANDC__
#define  _M_IX86                                    300             /* make sure this
is defined for Intel CPUs */
#endif
#endif

#ifdef _M_IX86
#define          LittleEndian            1           /* e.g., 1 for Pentium, 0 for 68K */
#define          ALIGN32                             0               /*   need   dword
alignment? (no for Pentium) */
#else    /* non-Intel platforms */
#define          LittleEndian            0           /* (assume big endian */
#define          ALIGN32                             1               /*   (assume   need
alignment for non-Intel) */
#endif

#if LittleEndian
#define          Bswap(x)                        (x)             /*   NOP   for   little-endian
machines */
#define          ADDR_XOR                        0               /*   NOP   for   little-endian
machines */
#else
#define          Bswap(x)                        ((ROR(x,8) & 0xFF00FF00) | (ROL(x,8) &
0x00FF00FF))
#define          ADDR_XOR                        3               /* convert byte address in
dword */
#endif

/*       Macros for extracting bytes from dwords (correct for endianness) */
#define  _b(x,N)  (((BYTE *)&x)[((N) & 3) ^ ADDR_XOR]) /* pick bytes out of a dword */

#define          b0(x)                   _b(x,0)         /* extract LSB of DWORD */
#define          b1(x)                   _b(x,1)
```

```
#define		b2(x)			_b(x,2)
#define		b3(x)			_b(x,3)		/* extract MSB of DWORD */


////////////////////////////////////////////////////

// stdafx.h : 標準のシステム インクルード ファイル、
//		または参照回数が多く、かつあまり変更されない
//		プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__1CF304E5_DEE8_42A3_83B4_4982611E1C8E__INCLUDED_)
#define AFX_STDAFX_H__1CF304E5_DEE8_42A3_83B4_4982611E1C8E__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif
// !defined(AFX_STDAFX_H__1CF304E5_DEE8_42A3_83B4_4982611E1C8E__INCLUDED_)


////////////////////////////////////////////////////

/*****************************************************************************
	TABLE.H		-- Tables, macros, constants for Twofish S-boxes and MDS matrix

	Submitters:
		Bruce Schneier, Counterpane Systems
		Doug Whiting,	Hi/fn
		John Kelsey,	Counterpane Systems
		Chris Hall,		Counterpane Systems
		David Wagner,	UC Berkeley

	Code Author:		Doug Whiting,	Hi/fn

	Version	1.00		April 1998

	Copyright 1998, Hi/fn and Counterpane Systems.  All rights reserved.

	Notes:
		*	Tab size is set to 4 characters in this file
		*	These definitions should be used in optimized and unoptimized
			versions to insure consistency.

*****************************************************************************/

/* for computing subkeys */
#define  SK_STEP			0x02020202u
```

```
#define  SK_BUMP                        0x01010101u
#define  SK_ROTL                        9

/* Reed-Solomon code parameters: (12,8) reversible code
        g(x) = x**4 + (a + 1/a) x**3 + a x**2 + (a + 1/a) x + 1
    where a = primitive root of field generator 0x14D */
#define  RS_GF_FDBK             0x14D           /* field generator */
#define  RS_rem(x)              ¥
        { BYTE  b  = (BYTE) (x >> 24);
                                        ¥
          DWORD g2 = ((b << 1) ^ ((b & 0x80) ? RS_GF_FDBK : 0 )) & 0xFF;          ¥
          DWORD g3 = ((b >> 1) & 0x7F) ^ ((b & 1) ? RS_GF_FDBK >> 1 : 0 ) ^ g2 ; ¥
          x = (x << 8) ^ (g3 << 24) ^ (g2 << 16) ^ (g3 << 8) ^ b;
         ¥
        }

/*      Macros for the MDS matrix
*       The MDS matrix is (using primitive polynomial 169):
*       01  EF  5B  5B
*       5B  EF  EF  01
*       EF  5B  01  EF
*       EF  01  EF  5B
*-----------------------------------------------------------------
* More statistical properties of this matrix (from MDS.EXE output):
*
* Min Hamming weight (one byte difference) =  8. Max=26.  Total =  1020.
* Prob[8]:      7     23     42     20     52     95     88     94    121    128     91
*             102     76     41     24      8      4      1      3      0      0      0
* Runs[8]:      2      4      5      6      7      8      9     11
* MSBs[8]:      1      4     15      8     18     38     40     43
* HW= 8: 05040705 0A080E0A 14101C14 28203828 50407050 01499101 A080E0A0
* HW= 9: 04050707 080A0E0E 10141C1C 20283838 40507070 80A0E0E0 C6432020 07070504
*        0E0E0A08 1C1C1410 38382820 70705040 E0E0A080 202043C6 05070407 0A0E080E
*        141C101C 28382038 50704070 A0E080E0 4320C620 02924B02 089A4508
* Min Hamming weight (two byte difference) =  3. Max=28.  Total = 390150.
* Prob[3]:      7     18     55    149    270    914   2185   5761 11363 20719 32079
*             43492 51612 53851 52098 42015 31117 20854 11538   6223   2492   1033
* MDS OK, ROR:   6+   7+   8+   9+ 10+ 11+ 12+ 13+ 14+ 15+ 16+
*                  17+ 18+ 19+ 20+ 21+ 22+ 23+ 24+ 25+ 26+
*/
#define  MDS_GF_FDBK            0x169   /* primitive polynomial for GF(256)*/
#define  LFSR1(x) ( ((x) >> 1)  ^ (((x) & 0x01) ?    MDS_GF_FDBK/2 : 0))
#define  LFSR2(x) ( ((x) >> 2)  ^ (((x) & 0x02) ?    MDS_GF_FDBK/2 : 0)  ¥
                                                          ^     (((x)     &     0x01)     ?
MDS_GF_FDBK/4 : 0))

#define  Mx_1(x) ((DWORD)  (x))             /* force result to dword so << will work */
#define  Mx_X(x) ((DWORD) ((x) ^               LFSR2(x)))   /* 5B */
#define  Mx_Y(x) ((DWORD) ((x) ^ LFSR1(x) ^ LFSR2(x)))   /* EF */

#define  M00            Mul_1
#define  M01            Mul_Y
#define  M02            Mul_X
#define  M03            Mul_X

#define  M10            Mul_X
#define  M11            Mul_Y
#define  M12            Mul_Y
#define  M13            Mul_1
```

```
#define  M20            Mul_Y
#define  M21            Mul_X
#define  M22            Mul_1
#define  M23            Mul_Y

#define  M30            Mul_Y
#define  M31            Mul_1
#define  M32            Mul_Y
#define  M33            Mul_X

#define  Mul_1   Mx_1
#define  Mul_X   Mx_X
#define  Mul_Y   Mx_Y

/*      Define the fixed p0/p1 permutations used in keyed S-box lookup.
        By changing the following constant definitions for P_ij, the S-boxes will
        automatically get changed in all the Twofish source code. Note that P_i0 is
        the "outermost" 8x8 permutation applied.    See the f32() function to see
        how these constants are to be    used.
*/
#define  P_00    1                                      /* "outermost" permutation */
#define  P_01    0
#define  P_02    0
#define  P_03    (P_01^1)                       /* "extend" to larger key sizes */
#define  P_04    1

#define  P_10    0
#define  P_11    0
#define  P_12    1
#define  P_13    (P_11^1)
#define  P_14    0

#define  P_20    1
#define  P_21    1
#define  P_22    0
#define  P_23    (P_21^1)
#define  P_24    0

#define  P_30    0
#define  P_31    1
#define  P_32    1
#define  P_33    (P_31^1)
#define  P_34    1

#define  p8(N)   P8x8[P_##N]                    /* some syntax shorthand */

/* fixed 8x8 permutation S-boxes */

/*******************************************************************
*   07:07:14   05/30/98   [4x4]   TestCnt=256. keySize=128. CRC=4BD14D9E.
* maxKeyed:   dpMax = 18. lpMax =100. fixPt =   8. skXor =   0. skDup =   6.
* log2(dpMax[ 6..18])=     --- 15.42   1.33   0.89   4.05   7.98 12.05
* log2(lpMax[ 7..12])=   9.32   1.01   1.16   4.23   8.02 12.45
* log2(fixPt[ 0.. 8])=   1.44   1.44   2.44   4.06   6.01   8.21 11.07 14.09 17.00
* log2(skXor[ 0.. 0])
* log2(skDup[ 0.. 6])=     ---   2.37   0.44   3.94   8.36 13.04 17.99
*******************************************************************/
CONST BYTE P8x8[2][256]=
```

```
        {
/*  p0:  */
/*  dpMax       = 10.  lpMax       = 64.  cycleCnt=   1  1  1  0.          */
/* 817D6F320B59ECA4.ECB81235F4A6709D.BA5E6D90C8F32471.D7F4126E9B3085CA. */
/* Karnaugh maps:
*  0111 0001 0011 1010. 0001 1001 1100 1111. 1001 1110 0011 1110. 1101 0101 1111 1001.
*  0101 1111 1100 0100. 1011 0101 0010 0000. 0101 1000 1100 0101. 1000 0111 0011 0010.
*  0000 1001 1110 1101. 1011 1000 1010 0011. 0011 1001 0101 0000. 0100 0010 0101 1011.
*  0111 0100 0001 0110. 1000 1011 1110 1001. 0011 0011 1001 1101. 1101 0101 0000 1100.
*/
        {
        0xA9, 0x67, 0xB3, 0xE8, 0x04, 0xFD, 0xA3, 0x76,
        0x9A, 0x92, 0x80, 0x78, 0xE4, 0xDD, 0xD1, 0x38,
        0x0D, 0xC6, 0x35, 0x98, 0x18, 0xF7, 0xEC, 0x6C,
        0x43, 0x75, 0x37, 0x26, 0xFA, 0x13, 0x94, 0x48,
        0xF2, 0xD0, 0x8B, 0x30, 0x84, 0x54, 0xDF, 0x23,
        0x19, 0x5B, 0x3D, 0x59, 0xF3, 0xAE, 0xA2, 0x82,
        0x63, 0x01, 0x83, 0x2E, 0xD9, 0x51, 0x9B, 0x7C,
        0xA6, 0xEB, 0xA5, 0xBE, 0x16, 0x0C, 0xE3, 0x61,
        0xC0, 0x8C, 0x3A, 0xF5, 0x73, 0x2C, 0x25, 0x0B,
        0xBB, 0x4E, 0x89, 0x6B, 0x53, 0x6A, 0xB4, 0xF1,
        0xE1, 0xE6, 0xBD, 0x45, 0xE2, 0xF4, 0xB6, 0x66,
        0xCC, 0x95, 0x03, 0x56, 0xD4, 0x1C, 0x1E, 0xD7,
        0xFB, 0xC3, 0x8E, 0xB5, 0xE9, 0xCF, 0xBF, 0xBA,
        0xEA, 0x77, 0x39, 0xAF, 0x33, 0xC9, 0x62, 0x71,
        0x81, 0x79, 0x09, 0xAD, 0x24, 0xCD, 0xF9, 0xD8,
        0xE5, 0xC5, 0xB9, 0x4D, 0x44, 0x08, 0x86, 0xE7,
        0xA1, 0x1D, 0xAA, 0xED, 0x06, 0x70, 0xB2, 0xD2,
        0x41, 0x7B, 0xA0, 0x11, 0x31, 0xC2, 0x27, 0x90,
        0x20, 0xF6, 0x60, 0xFF, 0x96, 0x5C, 0xB1, 0xAB,
        0x9E, 0x9C, 0x52, 0x1B, 0x5F, 0x93, 0x0A, 0xEF,
        0x91, 0x85, 0x49, 0xEE, 0x2D, 0x4F, 0x8F, 0x3B,
        0x47, 0x87, 0x6D, 0x46, 0xD6, 0x3E, 0x69, 0x64,
        0x2A, 0xCE, 0xCB, 0x2F, 0xFC, 0x97, 0x05, 0x7A,
        0xAC, 0x7F, 0xD5, 0x1A, 0x4B, 0x0E, 0xA7, 0x5A,
        0x28, 0x14, 0x3F, 0x29, 0x88, 0x3C, 0x4C, 0x02,
        0xB8, 0xDA, 0xB0, 0x17, 0x55, 0x1F, 0x8A, 0x7D,
        0x57, 0xC7, 0x8D, 0x74, 0xB7, 0xC4, 0x9F, 0x72,
        0x7E, 0x15, 0x22, 0x12, 0x58, 0x07, 0x99, 0x34,
        0x6E, 0x50, 0xDE, 0x68, 0x65, 0xBC, 0xDB, 0xF8,
        0xC8, 0xA8, 0x2B, 0x40, 0xDC, 0xFE, 0x32, 0xA4,
        0xCA, 0x10, 0x21, 0xF0, 0xD3, 0x5D, 0x0F, 0x00,
        0x6F, 0x9D, 0x36, 0x42, 0x4A, 0x5E, 0xC1, 0xE0
        },
/*  p1:  */
/*  dpMax       = 10.  lpMax       = 64.  cycleCnt=   2  0  0  1.          */
/* 28BDF76E31940AC5.1E2B4C376DA5F908.4C75169A0ED82B3F.B951C3DE647F208A. */
/* Karnaugh maps:
*  0011 1001 0010 0111. 1010 0111 0100 0110. 0011 0001 1111 0100. 1111 1000 0001 1100.
*  1100 1111 1111 1010. 0011 0011 1110 0100. 1001 0110 0100 0011. 0101 0110 1011 1011.
*  0010 0100 0011 0101. 1100 1000 1000 1110. 0111 1111 0010 0110. 0000 1010 0000 0011.
*  1101 1000 0010 0001. 0110 1001 1110 0101. 0001 0100 0101 0111. 0011 1011 1111 0010.
*/
        {
        0x75, 0xF3, 0xC6, 0xF4, 0xDB, 0x7B, 0xFB, 0xC8,
        0x4A, 0xD3, 0xE6, 0x6B, 0x45, 0x7D, 0xE8, 0x4B,
        0xD6, 0x32, 0xD8, 0xFD, 0x37, 0x71, 0xF1, 0xE1,
        0x30, 0x0F, 0xF8, 0x1B, 0x87, 0xFA, 0x06, 0x3F,
        0x5E, 0xBA, 0xAE, 0x5B, 0x8A, 0x00, 0xBC, 0x9D,
```

```
0x6D, 0xC1, 0xB1, 0x0E, 0x80, 0x5D, 0xD2, 0xD5,
0xA0, 0x84, 0x07, 0x14, 0xB5, 0x90, 0x2C, 0xA3,
0xB2, 0x73, 0x4C, 0x54, 0x92, 0x74, 0x36, 0x51,
0x38, 0xB0, 0xBD, 0x5A, 0xFC, 0x60, 0x62, 0x96,
0x6C, 0x42, 0xF7, 0x10, 0x7C, 0x28, 0x27, 0x8C,
0x13, 0x95, 0x9C, 0xC7, 0x24, 0x46, 0x3B, 0x70,
0xCA, 0xE3, 0x85, 0xCB, 0x11, 0xD0, 0x93, 0xB8,
0xA6, 0x83, 0x20, 0xFF, 0x9F, 0x77, 0xC3, 0xCC,
0x03, 0x6F, 0x08, 0xBF, 0x40, 0xE7, 0x2B, 0xE2,
0x79, 0x0C, 0xAA, 0x82, 0x41, 0x3A, 0xEA, 0xB9,
0xE4, 0x9A, 0xA4, 0x97, 0x7E, 0xDA, 0x7A, 0x17,
0x66, 0x94, 0xA1, 0x1D, 0x3D, 0xF0, 0xDE, 0xB3,
0x0B, 0x72, 0xA7, 0x1C, 0xEF, 0xD1, 0x53, 0x3E,
0x8F, 0x33, 0x26, 0x5F, 0xEC, 0x76, 0x2A, 0x49,
0x81, 0x88, 0xEE, 0x21, 0xC4, 0x1A, 0xEB, 0xD9,
0xC5, 0x39, 0x99, 0xCD, 0xAD, 0x31, 0x8B, 0x01,
0x18, 0x23, 0xDD, 0x1F, 0x4E, 0x2D, 0xF9, 0x48,
0x4F, 0xF2, 0x65, 0x8E, 0x78, 0x5C, 0x58, 0x19,
0x8D, 0xE5, 0x98, 0x57, 0x67, 0x7F, 0x05, 0x64,
0xAF, 0x63, 0xB6, 0xFE, 0xF5, 0xB7, 0x3C, 0xA5,
0xCE, 0xE9, 0x68, 0x44, 0xE0, 0x4D, 0x43, 0x69,
0x29, 0x2E, 0xAC, 0x15, 0x59, 0xA8, 0x0A, 0x9E,
0x6E, 0x47, 0xDF, 0x34, 0x35, 0x6A, 0xCF, 0xDC,
0x22, 0xC9, 0xC0, 0x9B, 0x89, 0xD4, 0xED, 0xAB,
0x12, 0xA2, 0x0D, 0x52, 0xBB, 0x02, 0x2F, 0xA9,
0xD7, 0x61, 0x1E, 0xB4, 0x50, 0x04, 0xF6, 0xC2,
0x16, 0x25, 0x86, 0x56, 0x55, 0x09, 0xBE, 0x91
}
};
```

１０．３．２　復号化部分

```
// DCTwofish.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include"aes.h"
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include<ctype.h>

extern   CONST char *moduleDescription; /* which module is running */
extern   CONST char *modeString;                              /* which key schedule mode */
extern   CONST int   debugCompile;                    /* is external module compiled with debug?
*/

#if defined(__WATCOMC__) && defined(_M_IX86) && !defined(NO_TIMER)
     DWORD ReadTimeStampCounter(void);
          #pragma aux ReadTimeStampCounter = " db 0Fh,031h" value [eax] modify exact [eax
edx] // RDTSC opcode
#endif


/*
+*****************************************************************************
```

```
*                               Constants/Macros/Tables
-************************************************************************/

typedef struct
        {
        FILE *f;                        /* the file being written/read */
        int   I;                                /* test number */
        int         keySize;                    /* key size in bits */
        int         gotDebugIO;             /* got any debug IO? */
        BYTE pt[BLOCK_SIZE/8];          /* plaintext */
        BYTE ct[BLOCK_SIZE/8];          /* ciphertext */

        keyInstance     ki;             /* use ki.keyDwords as key bits */
        cipherInstance ci;              /* use ci.iv as iv bits */
        } testData;


static char hexTab[]        =       "0123456789ABCDEF";
char                filePath[80]=       "";

int                         useAsm          =       0;      /* use assembly language */
int                         mctInner        =       MCT_INNER/100;
int                         mctOuter        =       MCT_OUTER/10;
int                         verify          =       0;      /* set to nonzero to read&verify
files */
int                         debug           =       0;      /* debugging mode */
int                         verbose         =       0;      /* verbose output */
int                         quietVerify     =       0;      /* quiet during verify */
int                         timeIterCnt     =       0;      /* how many times to iterate for
timing */
DWORD           randBits[64]= {1};      /* use Knuth's additive generator */
int                         randPtr;
testData *      debugTD                 = NULL;     /* for use with debugIO */
int                         CLKS_BYTE   =       0;      /* use clks/byte? (vs. clks/block) */
int                         FMT_LOG             =       0;      /* format for log file */
int                         CLK_MHZ             =       200;/* default clock speed */

#define             KEY_BITS_0                  128                     /* first key bit
setting to test */
#define             STEP_KEY_BITS               ((MAX_KEY_BITS-KEY_BITS_0)/2)
/*
static char   hexString[]=

        "0123456789ABCDEFFEDCBA98765432100011223344556677889AABBCCDDEEFF"
;
*/
static char   hex7String[72];// use 64 byte
//=
        "123456712345671234567123456712345671234567123456712345671234567123456712345671";
////////////////////////////////////////////
/*
+************************************************************************
*                               Functions
-************************************************************************/

DWORD Here(DWORD x)
        {
        unsigned int mask=~0U;
```

```
        return (* (((DWORD *)&x)-1)) & mask;
        }
extern DWORD TwofishCodeSize(void);

#ifdef USE_ASM
int cdecl get_cpu_type(void);                   /* return CPU type */
#endif

/*
+*****************************************************************************
*
* Function Name:        Rand
*
* Function:                     Generate random number
*
* Arguments:            None.
*
* Return:                       New random number.
*
* Notes:                Uses Knuth's additive generator, other magic
*
-*****************************************************************************/
DWORD Rand(void)
        {
        if (randPtr >= 57)
                randPtr = 0;                    /* handle the ptr wrap */

        randBits[randPtr] += randBits[(randPtr < 7) ? randPtr-7+57 : randPtr-7];

        randBits[62]+= randBits[61];
        randBits[63] = ROL(randBits[63],9) + 0x6F4ED7D0;/* very long period! */

        return (randBits[randPtr++] ^ randBits[63]) + randBits[62];
        }


/*
+*****************************************************************************
*
* Function Name:        SetRand
*
* Function:                     Initialize random number seed
*
* Arguments:            seed    =       new seed value
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void SetRand(DWORD seed)
        {
        int i;
        DWORD x;

        randPtr=0;
        for (i=x=0;i<64;i++)
                {
                randBits[i]=seed;
```

```
                x |= seed;                  /* keep track of lsb of all entries */
                seed = ROL(seed,11) + 0x12345678;
                }

        if ((x & 1) == 0)   /* insure maximal period by having at least one odd value */
                randBits[0]++;

        for (i=0;i<1000;i++)
                Rand();                     /* run it for a while */

        randBits[63] = Rand();
        randBits[62] = Rand();
        randBits[61] = Rand() | 1;/* make it odd */
        }


/*
+*****************************************************************************
*
* Function Name:        ClearTestData
*
* Function:                     Initialize test data to all zeroes
*
* Arguments:            t               =       pointer to testData structure
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void ClearTestData(testData *t)
        {
        t->gotDebugIO=0;
        memset(t->pt,0,BLOCK_SIZE/8);
        memset(t->ct,0,BLOCK_SIZE/8);
        memset(t->ci.iv32,0,BLOCK_SIZE/8);
        memset(t->ki.key32,0,MAX_KEY_BITS/8);
        memset(t->ki.keyMaterial,'0',sizeof(t->ki.keyMaterial));
#if defined(COMPILE_KEY) && defined(USE_ASM)
        t->ki.cSig1=t->ki.cSig2=0;
#endif
        }

/*
+*****************************************************************************
*
* Function Name:        FatalError
*
* Function:                     Output a fatal error message and exit
*
* Arguments:            msg             =       fatal error description (printf string)
*                               msg2    =       2nd parameter to printf msg
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void FatalError(CONST char *msg,CONST char *msg2)
```

```
        {
        printf("¥nFATAL ERROR: ");
        printf(msg,msg2);
        exit(2);
        }


/*
+*****************************************************************************
*
* Function Name:        AES_FileIO
*
* Function:                     Output to file or verify file contents vs. string
*
* Arguments:            f               =       opened file
*                               s               =       string to output/compare
(NULL-->reset, return)
*                               errOK   =       do not fatalError on miscompare
*
* Return:                       Zero --> compare ok
*
* Notes:                        On miscompare, FatalError (unless errOK)
*
-*****************************************************************************/
int AES_FileIO(FILE *f,CONST char *s,int errOK)
        {
        int   i;
        static int   lineNum=0;
        static int   j=0;
        static char line[516]="";

        if (s == NULL)    /* starting new file */
                {
                line[0]=j=lineNum=0;
                return 0;
                }

        if (!verify)
                {
                fprintf(f,s);
                return 0;
                }

        /* here to verify the file against the string */
        for (i=0;s[i];i++)
                {
                while (line[j] == 0)
                        {
                        lineNum++;
                        if (fgets(line,sizeof(line)-4,f) == NULL)
                                {
                                if ((s[i]=='¥n') && (s[i+1]==0))
                                        {
                                        line[0]=j=0;        /* missing final eol is ok */
                                        return 0;
                                        }
                                FatalError("Unexpected EOF looking for %s",s);
                                }
                        if (verbose) printf(line);
```

```
                        j=0;
                        }
                if (s[i] != line[j])
                        {
                        if ((s[i] == '¥n') && ((i==0) || (s[i-1] == '¥n'))) continue; /* blank line
skip */
                        if (line[j] == '¥n') {j++; continue;}
                        if (!errOK)
                                {
                                char tmp[1024];
                                sprintf(tmp,"Miscompare at line #%d:¥n%s¥nlooking
for¥n¥n%%s",lineNum,line);
                                FatalError(tmp,s);
                                }
                        line[0]=j=0;    /* let caller re-synch if desired */
                        return 1;               /* return error flag */
                        }
                j++;
                }

        return 0;
        }



/*
+*************************************************************************
*
* Function Name:        DebugIO
*
* Function:                     Output debug string
*
* Arguments:            s       =       string to output
*
* Return:                       None.
*
* Notes:
*
-*************************************************************************/
void DebugIO(CONST char *s)
        {
        if (debugTD)
                {
                AES_FileIO(debugTD->f,s,0);
                debugTD->gotDebugIO=1;
                }
        else
                printf(s);
        }


//////////////////////////////////////////
//////////////////////////////////////////////////////////////
/*************************************************************/
////////////   uyama
void UY_DC_TF(char *key, char *ct, char *pt){
        FILE *fkey;
        FILE *stream1;
```

```
        FILE *stream2;
        char c_mode[8], c_klen[8], pass[128];
        int i,mode,klen;
        testData t;
        int head;
        long lenp;
        int cc, block;
        char* bufc;
        unsigned char ostr[BLOCK_SIZE/8+10];
        // 暗号文格納場所へのポインタ

        printf("TfDC Start!¥n" );
        /* 鍵を読み出すファイルを開く */
        if( (fkey   = fopen( key, "rt" )) == NULL ){
                printf( "Can not open key file.¥n");
                return;
        }
        fgets( c_mode, 6, fkey );
        fgets( c_klen, 6, fkey );
        fgets( pass, 127, fkey );
        mode = atoi(c_mode);
        klen = atoi(c_klen);
    /* 鍵*/
        for(i=0; i<klen/4; i++){
                hex7String[i] = pass[i];
        }
        hex7String[klen/4] = NULL;

        /* 平文を書き込むファイルを開く */
    if( (stream1   = fopen(pt , "wb" )) == NULL )
        printf( "Can not open plane text file.¥n");
    /* 暗号文を読み出すファイルを開く  */
    if( (stream2 = fopen( ct, "rb" )) == NULL )
        printf( "Can not open file for encrypted data.¥n" );

////////////////////////////////////////////////////////////////
    // 暗文
        fseek(stream2, 0, SEEK_END);
        long filelen = ftell(stream2);
        fseek(stream2,0,0);
        head = sizeof(long);

        t.keySize=klen;
        if (cipherInit(&t.ci,mode,hex7String) != TRUE)
                FatalError("cipherInit error during %s test",""/*fname*/);
        ClearTestData(&t);                                      /* start with all zeroes */
        if (makeKey(&t.ki,DIR_DECRYPT,t.keySize,hex7String/*t.ki.keyMaterial*/) != TRUE)
                FatalError("Error parsing key during %s test",""/*fname*/);

        //暗号化
        if(filelen <= BLOCK_SIZE/8){//63 が暗号化の作業サイズ　63*20＝1260
                bufc = (char*)new(char[BLOCK_SIZE/8+10]);
                int i=0;
                if(bufc == NULL){
                        printf("No memory");
                        return;
                }
                do{
                        cc = fgetc(stream2);
```

```
                        bufc[i]=cc;
                        i=i+1;
                }while(cc!=EOF);
                bufc[i-1]=NULL;
                for(int j=0; j+i<BLOCK_SIZE/8+10; j++){
                        bufc[j+i] = NULL;
                }

                memcpy(t.ct,bufc,BLOCK_SIZE/8);
    // 復号化実行
                if (blockDecrypt(&t.ci,&t.ki,t.ct,BLOCK_SIZE,t.pt) != BLOCK_SIZE)
                        FatalError("blockDecrypt return during %s test","ff.bin"/*fname*/);
                memcpy(ostr,t.pt,BLOCK_SIZE/8);
                lenp = *((long*)ostr);
                fseek(stream1,0,0);
                fwrite( ostr+head,sizeof(char),lenp, stream1);
                fclose(stream1);
        }
        else{
                long rBlen = filelen;
                bufc = (char*)new(char[BLOCK_SIZE/8+10]);//63 が暗号化の作業サイズ　63*20
＝1260
                if(bufc == NULL){
                        printf("メモリ不足¥r¥n");
                        return;
                }
                int r = 0;
                do{
                        fseek(stream2, r*BLOCK_SIZE/8, 0);//63 が暗号化の作業サイズ
63*20＝1260
                        i=0;
                        do{
                                cc = fgetc(stream2);
                                bufc[i]=cc;
                                i=i+1;
                        }while((cc!=EOF) && (i<=BLOCK_SIZE/8));//63 が暗号化の作業サイズ
63*20＝1260
                        bufc[i-1]=NULL;
                        for(int j=0; j+i<BLOCK_SIZE/8+10; j++){
                                bufc[j+i] = NULL;
                        }

                        if(rBlen >= BLOCK_SIZE/8){ block = BLOCK_SIZE/8; }//63 が暗号化の
作業サイズ　63*20＝1260
                        if(rBlen < BLOCK_SIZE/8){ block = rBlen;}

                        memcpy(t.ct,bufc,BLOCK_SIZE/8);
    // 復号化実行
                        if (blockDecrypt(&t.ci,&t.ki,t.ct,BLOCK_SIZE,t.pt) != BLOCK_SIZE)
                                FatalError("blockDecrypt return during %s
test","ff.bin"/*fname*/);
                        memcpy(ostr,t.pt,BLOCK_SIZE/8);
        // 復号文を書き込む
                        if(r==0){
                                int head = sizeof(long);
                                lenp = *((long*)ostr);
                                int wc = fwrite( ostr+head,sizeof(char),BLOCK_SIZE/8-head,
stream1);
                                lenp -= BLOCK_SIZE/8-head;
```

```
                        }
                        if(r>=1){
                                if(lenp >= BLOCK_SIZE/8){
                                        fwrite( ostr,sizeof(char),BLOCK_SIZE/8, stream1);
                                        lenp -= BLOCK_SIZE/8;
                                }
                                else {
                                        if(lenp < BLOCK_SIZE/8){
                                                fwrite( ostr,sizeof(char),lenp, stream1);
                                                lenp -= lenp;
                                        }
                                }
                        }
                        r += 1;
                        rBlen -= block;
                }while(rBlen>0);
        }
        _fcloseall();
        printf("TfDC End!¥n" );
}

/*****************************************************************/


/*
+*****************************************************************************
*
* Function Name:        GiveHelp
*
* Function:                     Print out list of command line switches
*
* Arguments:            None.
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void GiveHelp(void)
        {
        printf("Syntax:    TST2FISH [options]¥n"
                "Purpose:   Generate/validate AES Twofish code and files¥n"
                "Options:   -lNN      ==> set sanity check loop to NN¥n"
                "           -m        ==> do full MCT generation¥n"
                "           -pPath    ==> set file path¥n"
                "           -s        ==> set initial random seed based on time¥n"
                "           -sNN      ==> set initial random seed to NN¥n"
                "           -tNN      ==> time performance using NN iterations¥n"
                "           -v        ==> validate files, don't generate them¥n",
                MAX_ROUNDS
               );
        exit(1);
        }

#ifdef TEST_EXTERN
void    Test_Extern(void);
#endif

void ShowHex(FILE *f,CONST void *p,int bCnt,CONST char *name)
```

```
        {
        int i;

        fprintf(f,"      ;%s:",name);
        for (i=0;i<bCnt;i++)
                {
                if ((i % 8) == 0)
                        fprintf(f,"¥n¥t.byte¥t");
                else
                        fprintf(f,",");
                fprintf(f,"0%02Xh",((BYTE *)p)[i]);
                }
        fprintf(f,"¥n");
        }

/* output a formatted 6805 test vector include file */
void Debug6805(void)
        {
        int i,j;
        testData t;
        FILE *f;

        ClearTestData(&t);
        t.keySize=128;

        f=stdout;
        cipherInit(&t.ci,MODE_ECB,NULL);
        makeKey(&t.ki,DIR_ENCRYPT,t.keySize,t.ki.keyMaterial);

        for (i=0;i<4;i++)   /* make sure it all fits in 256 bytes */
                {
                reKey(&t.ki);
                blockEncrypt(&t.ci,&t.ki,t.pt,BLOCK_SIZE,t.ct);
                fprintf(f,"; Twofish vector #%d¥n",i+1);
                ShowHex(f,&t.keySize,1,"Key Size");
                ShowHex(f,t.ki.key32,16,"Key");
                ShowHex(f,t.pt,BLOCK_SIZE/8,"Plaintext");
                ShowHex(f,t.ct,BLOCK_SIZE/8,"Ciphertext");
                for (j=0;j<16;j++)
                        ((BYTE *)t.ki.key32)[j] = t.pt[j] ^ t.ct[j];
                memcpy(t.pt,t.ct,sizeof(t.pt));
                fprintf(f,";-----------------------------------------------------------¥n");
                }
        fprintf(f,"¥n¥t.byte 0¥t;end of list¥n");
        fclose(f);
        }


//////////////////////////////////////////////
int main(int argc, char* argv[])
{
#define  MAX_ARGS        40
        int i,testCnt=32;
        DWORD randSeed=0x12345678;
        char *moduleName=moduleDescription;

        // 引数チェック
        if( argc != 4 ){                                        // 使い方の誤り
                exit(1);
```

```
        }

        i=1;                            /* make sure LittleEndian is defined correctly */
        if (b0(i) != 1)
                FatalError("LittleEndian defined incorrectly","");
//      if ((ALIGN32) && (k == 2))
//              FatalError("Cannot enable ALIGN32 in 16-bit mode¥n","");

#if ((MCT_INNER != 10000) || (MCT_OUTER != 400))
#error  MCT loop counts incorrect!
#endif

#ifdef USE_ASM
        if (useAsm & 7)   moduleName="Assembler ";
#endif

        SetRand(randSeed);                                              /*    init    pseudorandom
generator for testing */

#ifdef TEST_EXTERN
        Test_Extern();
        exit(0);
#endif

        UY_DC_TF(argv[1],argv[2],argv[3]);

        return 0;
        }

////////////////////////////////////////////////////////////////////////////

// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//              DCTwofish.pch 生成されるプリコンパイル済ヘッダー
//              stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。

////////////////////////////////////////////////////////

/*****************************************************************************
        TWOFISH2.C      -- Optimized C API calls for TWOFISH AES submission

        Submitters:
                Bruce Schneier, Counterpane Systems
                Doug Whiting,   Hi/fn
                John Kelsey,    Counterpane Systems
                Chris Hall,             Counterpane Systems
                David Wagner,   UC Berkeley

        Code Author:            Doug Whiting,   Hi/fn

        Version  1.00           April 1998

        Copyright 1998, Hi/fn and Counterpane Systems.  All rights reserved.

        Notes:
```

```
                    *       Optimized version
                    *       Tab size is set to 4 characters in this file

*************************************************************************/
#include "stdafx.h"

#include"aes.h"
#include"table.h"

#include<memory.h>
#include<assert.h>

#define GetCodeSize //Uyama

#if     defined(min_key)    && !defined(MIN_KEY)
#define  MIN_KEY                    1                           /* toupper() */
#elif defined(part_key) && !defined(PART_KEY)
#define  PART_KEY       1
#elif defined(zero_key) && !defined(ZERO_KEY)
#define  ZERO_KEY       1
#endif


#ifdef USE_ASM
extern    int       useAsm;                         /* ok to use ASM code? */

typedef  int cdecl CipherProc
    (cipherInstance *cipher, keyInstance *key,BYTE *input,int inputLen,BYTE *outBuffer);
typedef int         cdecl KeySetupProc(keyInstance *key);

extern CipherProc       *blockEncrypt_86;       /* ptr to ASM functions */
extern CipherProc       *blockDecrypt_86;
extern KeySetupProc     *reKey_86;
extern DWORD            cdecl TwofishAsmCodeSize(void);
#endif

/*
+*****************************************************************************
*                           Constants/Macros/Tables
-*****************************************************************************/

#define             CONST                                   /* help syntax from C++, NOP here
*/

CONST               fullSbox MDStab;                /* not actually const.    Initialized ONE time
*/
int                         needToBuildMDS=1;               /* is MDStab initialized yet? */

#define             BIG_TAB                 0

#if BIG_TAB
BYTE                bigTab[4][256][256];        /* pre-computed S-box */
#endif

/* number of rounds for various key sizes:    128, 192, 256 */
/* (ignored for now in optimized code!) */
CONST int           numRounds[4]= {0,ROUNDS_128,ROUNDS_192,ROUNDS_256};

#if REENTRANT
```

```
#define             _sBox_    key->sBox8x32
#else
static              fullSbox _sBox_;            /* permuted MDStab based on keys */
#endif
#define _sBox8_(N) (((BYTE *) _sBox_) + (N)*256)

/*------- see what level of S-box precomputation we need to do -----*/
#if    defined(ZERO_KEY)
#define  MOD_STRING   "(Zero S-box keying)"
#define  Fe32_128(x,R)    ¥
         (          MDStab[0][p8(01)[p8(02)[_b(x,R   )]^b0(SKEY[1])]^b0(SKEY[0])] ^    ¥
                    MDStab[1][p8(11)[p8(12)[_b(x,R+1)]^b1(SKEY[1])]^b1(SKEY[0])] ^    ¥
                    MDStab[2][p8(21)[p8(22)[_b(x,R+2)]^b2(SKEY[1])]^b2(SKEY[0])] ^    ¥
                    MDStab[3][p8(31)[p8(32)[_b(x,R+3)]^b3(SKEY[1])]^b3(SKEY[0])] )
#define  Fe32_192(x,R)    ¥
         (
         MDStab[0][p8(01)[p8(02)[p8(03)[_b(x,R    )]^b0(SKEY[2])]^b0(SKEY[1])]^b0(SKEY[0])]
^ ¥

         MDStab[1][p8(11)[p8(12)[p8(13)[_b(x,R+1)]^b1(SKEY[2])]^b1(SKEY[1])]^b1(SKEY[0])]
^ ¥

         MDStab[2][p8(21)[p8(22)[p8(23)[_b(x,R+2)]^b2(SKEY[2])]^b2(SKEY[1])]^b2(SKEY[0])]
^ ¥

         MDStab[3][p8(31)[p8(32)[p8(33)[_b(x,R+3)]^b3(SKEY[2])]^b3(SKEY[1])]^b3(SKEY[0])]
)
#define  Fe32_256(x,R)    ¥
         (
         MDStab[0][p8(01)[p8(02)[p8(03)[p8(04)[_b(x,R    )]^b0(SKEY[3])]^b0(SKEY[2])]^b0(SK
EY[1])]^b0(SKEY[0])] ^ ¥

         MDStab[1][p8(11)[p8(12)[p8(13)[p8(14)[_b(x,R+1)]^b1(SKEY[3])]^b1(SKEY[2])]^b1(SK
EY[1])]^b1(SKEY[0])] ^ ¥

         MDStab[2][p8(21)[p8(22)[p8(23)[p8(24)[_b(x,R+2)]^b2(SKEY[3])]^b2(SKEY[2])]^b2(SK
EY[1])]^b2(SKEY[0])] ^ ¥

         MDStab[3][p8(31)[p8(32)[p8(33)[p8(34)[_b(x,R+3)]^b3(SKEY[3])]^b3(SKEY[2])]^b3(SK
EY[1])]^b3(SKEY[0])] )

#define  GetSboxKey       DWORD SKEY[4];          /* local copy */ ¥
                                                  memcpy(SKEY,key->sboxKeys,sizeof(SKEY));
/*-------------------------------------------------------------*/
#elif defined(MIN_KEY)
#define  MOD_STRING   "(Minimal keying)"
#define  Fe32_(x,R)(MDStab[0][p8(01)[_sBox8_(0)[_b(x,R   )]] ^ b0(SKEY0)] ^ ¥
                                        MDStab[1][p8(11)[_sBox8_(1)[_b(x,R+1)]] ^ b1(SKEY0)] ^
¥
                                        MDStab[2][p8(21)[_sBox8_(2)[_b(x,R+2)]] ^ b2(SKEY0)] ^
¥
                                        MDStab[3][p8(31)[_sBox8_(3)[_b(x,R+3)]] ^ b3(SKEY0)])
#define sbSet(N,i,J,v) { _sBox8_(N)[i+J] = v; }
#define  GetSboxKey       DWORD SKEY0 = key->sboxKeys[0]                  /* local copy */
/*-------------------------------------------------------------*/
#elif defined(PART_KEY)
#define  MOD_STRING   "(Partial keying)"
#define  Fe32_(x,R)(MDStab[0][_sBox8_(0)[_b(x,R   )]] ^ ¥
                                        MDStab[1][_sBox8_(1)[_b(x,R+1)]] ^ ¥
```

```
                                        MDStab[2][_sBox8_(2)[_b(x,R+2)]] ^ ¥
                                        MDStab[3][_sBox8_(3)[_b(x,R+3)]])
#define sbSet(N,i,J,v) { _sBox8_(N)[i+J] = v; }
#define  GetSboxKey
/*-----------------------------------------------------------------*/
#else     /* default is FULL_KEY */
#ifndef FULL_KEY
#define  FULL_KEY       1
#endif
#if BIG_TAB
#define  TAB_STR                " (Big table)"
#else
#define  TAB_STR
#endif
#ifdef COMPILE_KEY
#define  MOD_STRING   "(Compiled subkeys)" TAB_STR
#else
#define  MOD_STRING   "(Full keying)" TAB_STR
#endif
/* Fe32_ does a full S-box + MDS lookup.   Need to #define _sBox_ before use.
    Note that we "interleave" 0,1, and 2,3 to avoid cache bank collisions
    in optimized assembly language.
*/
#define  Fe32_(x,R) (_sBox_[0][2*_b(x,R   )] ^ _sBox_[0][2*_b(x,R+1)+1] ^       ¥
                                        _sBox_[2][2*_b(x,R+2)] ^ _sBox_[2][2*_b(x,R+3)+1])
                /* set a single S-box value, given the input byte */
#define sbSet(N,i,J,v) { _sBox_[N&2][2*i+(N&1)+2*J]=MDStab[N][v]; }
#define  GetSboxKey
#endif

CONST           char *moduleDescription  ="Optimized C ";
CONST           char *modeString                =MOD_STRING;


/* macro(s) for debugging help */
#define         CHECK_TABLE             0               /*  nonzero --> compare against
"slow" table */
#define         VALIDATE_PARMS  0               /* disable for full speed */

#include"debug.h"                                       /* debug display macros */

/* end of debug macros */

#ifdef GetCodeSize
extern DWORD Here(DWORD x);                     /* return caller's address! */
DWORD TwofishCodeStart(void) { return Here(0); }
#endif

/*
+*****************************************************************************
*
* Function Name:        TableOp
*
* Function:                     Handle table use checking
*
* Arguments:            op      =       what to do      (see TAB_* defns in AES.H)
*
* Return:                       TRUE --> done (for TAB_QUERY)
*
```

```
* Notes: This routine is for use in generating the tables KAT file.
*                   For this optimized version, we don't actually track table usage,
*                   since it would make the macros incredibly ugly.    Instead we just
*                   run for a fixed number of queries and then say we're done.
*
-************************************************************************/
int TableOp(int op)
        {
        static int queryCnt=0;

        switch (op)
                {
                case TAB_DISABLE:
                        break;
                case TAB_ENABLE:
                        break;
                case TAB_RESET:
                        queryCnt=0;
                        break;
                case TAB_QUERY:
                        queryCnt++;
                        if (queryCnt < TAB_MIN_QUERY)
                                return FALSE;
                }
        return TRUE;
        }


/*
+************************************************************************
*
* Function Name:        ParseHexDword
*
* Function:                     Parse ASCII hex nibbles and fill in key/iv dwords
*
* Arguments:            bit                     =       # bits to read
*                                       srcTxt          =       ASCII source
*                                       d                       =       ptr to dwords to
fill in
*                                       dstTxt          =       where to make a copy of
ASCII source
*                                                                       (NULL ok)
*
* Return:                       Zero if no error.    Nonzero --> invalid hex or length
*
* Notes:    Note that the parameter d is a DWORD array, not a byte array.
*           This routine is coded to work both for little-endian and big-endian
*           architectures.    The character stream is interpreted as a LITTLE-ENDIAN
*           byte stream, since that is how the Pentium works, but the conversion
*           happens automatically below.
*
-************************************************************************/
int ParseHexDword(int bits,CONST char *srcTxt,DWORD *d,char *dstTxt)
        {
        int i;
        char c;
        DWORD b;

        union     /* make sure LittleEndian is defined correctly */
```

```
                {
                BYTE    b[4];
                DWORD d[1];
                } v;
        v.d[0]=1;
        if (v.b[0 ^ ADDR_XOR] != 1)
                return BAD_ENDIAN;              /* make sure compile-time switch is set ok */

#if VALIDATE_PARMS
  #if ALIGN32
        if (((int)d) & 3)
                return BAD_ALIGN32;
  #endif
#endif

        for (i=0;i*32<bits;i++)
                d[i]=0;                                 /* first, zero the field */

        for (i=0;i*4<bits;i++)          /* parse one nibble at a time */
                {                                               /*      case      out      the
hexadecimal characters */
                c=srcTxt[i];
                if (dstTxt) dstTxt[i]=c;
                if ((c >= '0') && (c <= '9'))
                        b=c-'0';
                else if ((c >= 'a') && (c <= 'f'))
                        b=c-'a'+10;
                else if ((c >= 'A') && (c <= 'F'))
                        b=c-'A'+10;
                else
                        return BAD_KEY_MAT;   /* invalid hex character */
                /* works for big and little endian! */
                d[i/8] |= b << (4*((i^1)&7));
                }

        return 0;                                       /* no error */
        }


#if CHECK_TABLE
/*
+*****************************************************************************
*
* Function Name:        f32
*
* Function:                     Run four bytes through keyed S-boxes and apply MDS matrix
*
* Arguments:            x                       =       input to f function
*                                       k32                     =       pointer    to    key
dwords
*                                       keyLen          =       total key length (k32 -->
keyLey/2 bits)
*
* Return:                       The output of the keyed permutation applied to x.
*
* Notes:
*       This function is a keyed 32-bit permutation.  It is the major building
*       block for the Twofish round function, including the four keyed 8x8
*       permutations and the 4x4 MDS matrix multiply.  This function is used
```

```
*       both for generating round subkeys and within the round function on the
*       block being encrypted.
*
*       This version is fairly slow and pedagogical, although a smartcard would
*       probably perform the operation exactly this way in firmware.    For
*       ultimate performance, the entire operation can be completed with four
*       lookups into four 256x32-bit tables, with three dword xors.
*
*       The MDS matrix is defined in TABLE.H.   To multiply by Mij, just use the
*       macro Mij(x).
*
-*****************************************************************************/
DWORD f32(DWORD x,CONST DWORD *k32,int keyLen)
        {
        BYTE    b[4];

        /* Run each byte thru 8x8 S-boxes, xoring with key byte at each stage. */
        /* Note that each byte goes through a different combination of S-boxes.*/

        *((DWORD *)b) = Bswap(x);           /* make b[0] = LSB, b[3] = MSB */
        switch (((keyLen + 63)/64) & 3)
                {
                case 0:                 /* 256 bits of key */
                        b[0] = p8(04)[b[0]] ^ b0(k32[3]);
                        b[1] = p8(14)[b[1]] ^ b1(k32[3]);
                        b[2] = p8(24)[b[2]] ^ b2(k32[3]);
                        b[3] = p8(34)[b[3]] ^ b3(k32[3]);
                        /* fall thru, having pre-processed b[0]..b[3] with k32[3] */
                case 3:                 /* 192 bits of key */
                        b[0] = p8(03)[b[0]] ^ b0(k32[2]);
                        b[1] = p8(13)[b[1]] ^ b1(k32[2]);
                        b[2] = p8(23)[b[2]] ^ b2(k32[2]);
                        b[3] = p8(33)[b[3]] ^ b3(k32[2]);
                        /* fall thru, having pre-processed b[0]..b[3] with k32[2] */
                case 2:                 /* 128 bits of key */
                        b[0] = p8(00)[p8(01)[p8(02)[b[0]] ^ b0(k32[1])] ^ b0(k32[0])];
                        b[1] = p8(10)[p8(11)[p8(12)[b[1]] ^ b1(k32[1])] ^ b1(k32[0])];
                        b[2] = p8(20)[p8(21)[p8(22)[b[2]] ^ b2(k32[1])] ^ b2(k32[0])];
                        b[3] = p8(30)[p8(31)[p8(32)[b[3]] ^ b3(k32[1])] ^ b3(k32[0])];
                }

        /* Now perform the MDS matrix multiply inline. */
        return  ((M00(b[0]) ^ M01(b[1]) ^ M02(b[2]) ^ M03(b[3]))        ) ^
                        ((M10(b[0]) ^ M11(b[1]) ^ M12(b[2]) ^ M13(b[3])) <<  8) ^
                        ((M20(b[0]) ^ M21(b[1]) ^ M22(b[2]) ^ M23(b[3])) << 16) ^
                        ((M30(b[0]) ^ M31(b[1]) ^ M32(b[2]) ^ M33(b[3])) << 24) ;
        }
#endif  /* CHECK_TABLE */


/*
+*****************************************************************************
*
* Function Name:        RS_MDS_encode
*
* Function:                     Use (12,8) Reed-Solomon code over GF(256) to produce
*                                       a key S-box dword from two key material dwords.
*
* Arguments:            k0      =       1st dword
```

```
*                                       k1      =       2nd dword
*
* Return:                       Remainder polynomial generated using RS code
*
* Notes:
*       Since this computation is done only once per reKey per 64 bits of key,
*       the performance impact of this routine is imperceptible. The RS code
*       chosen has "simple" coefficients to allow smartcard/hardware implementation
*       without lookup tables.
*
-*****************************************************************************/
DWORD RS_MDS_Encode(DWORD k0,DWORD k1)
        {
        int i,j;
        DWORD r;

        for (i=r=0;i<2;i++)
                {
                r ^= (i) ? k0 : k1;             /* merge in 32 more key bits */
                for (j=0;j<4;j++)               /* shift one byte at a time */
                        RS_rem(r);
                }
        return r;
        }


/*
+*****************************************************************************
*
* Function Name:        BuildMDS
*
* Function:                     Initialize the MDStab array
*
* Arguments:            None.
*
* Return:                       None.
*
* Notes:
*       Here we precompute all the fixed MDS table.  This only needs to be done
*       one time at initialization, after which the table is "CONST".
*
-*****************************************************************************/
void BuildMDS(void)
        {
        int i;
        DWORD d;
        BYTE m1[2],mX[2],mY[4];

        for (i=0;i<256;i++)
                {
                m1[0]=P8x8[0][i];       /* compute all the matrix elements */
                mX[0]=(BYTE) Mul_X(m1[0]);
                mY[0]=(BYTE) Mul_Y(m1[0]);

                m1[1]=P8x8[1][i];
                mX[1]=(BYTE) Mul_X(m1[1]);
                mY[1]=(BYTE) Mul_Y(m1[1]);

#undef  Mul_1                           /* change what the pre-processor does with
```

```
Mij */
#undef  Mul_X
#undef  Mul_Y
#define Mul_1   m1                              /* It will now access m01[], m5B[], and
mEF[] */
#define Mul_X   mX
#define Mul_Y   mY

#define SetMDS(N)                                               ¥
                b0(d) = M0##N[P_##N##0];        ¥
                b1(d) = M1##N[P_##N##0];        ¥
                b2(d) = M2##N[P_##N##0];        ¥
                b3(d) = M3##N[P_##N##0];        ¥
                MDStab[N][i] = d;

                SetMDS(0);                              /* fill in the matrix with elements
computed above */
                SetMDS(1);
                SetMDS(2);
                SetMDS(3);
                }
#undef  Mul_1
#undef  Mul_X
#undef  Mul_Y
#define Mul_1   Mx_1                    /* re-enable true multiply */
#define Mul_X   Mx_X
#define Mul_Y   Mx_Y

#if BIG_TAB
        {
        int j,k;
        BYTE *q0,*q1;

        for (i=0;i<4;i++)
                {
                switch (i)
                        {
                        case 0:   q0=p8(01); q1=p8(02);     break;
                        case 1:   q0=p8(11); q1=p8(12);     break;
                        case 2:   q0=p8(21); q1=p8(22);     break;
                        case 3:   q0=p8(31); q1=p8(32);     break;
                        }
                for (j=0;j<256;j++)
                        for (k=0;k<256;k++)
                                bigTab[i][j][k]=q0[q1[k]^j];
                }
        }
#endif

        needToBuildMDS=0;                       /* NEVER modify the table again! */
        }

/*
+*****************************************************************************
*
* Function Name:        ReverseRoundSubkeys
*
* Function:                     Reverse order of round subkeys to switch between
encrypt/decrypt
```

```
*
* Arguments:            key             =       ptr to keyInstance to be reversed
*                                       newDir  =       new direction value
*
* Return:                       None.
*
* Notes:
*       This optimization allows both blockEncrypt and blockDecrypt to use the same
*       "fallthru" switch statement based on the number of rounds.
*       Note that key->numRounds must be even and >= 2 here.
*
-*****************************************************************************/
void ReverseRoundSubkeys(keyInstance *key,BYTE newDir)
        {
        DWORD t0,t1;
        register DWORD *r0=key->subKeys+ROUND_SUBKEYS;
        register DWORD *r1=r0 + 2*key->numRounds - 2;

        for (;r0 < r1;r0+=2,r1-=2)
                {
                t0=r0[0];               /* swap the order */
                t1=r0[1];
                r0[0]=r1[0];            /* but keep relative order within pairs */
                r0[1]=r1[1];
                r1[0]=t0;
                r1[1]=t1;
                }

        key->direction=newDir;
        }

/*
+*****************************************************************************
*
* Function Name:        Xor256
*
* Function:                     Copy an 8-bit permutation (256 bytes), xoring with a byte
*
* Arguments:            dst             =       where to put result
*                                       src             =       where to get data (can be
same asa dst)
*                                       b               =       byte to xor
*
* Return:                       None
*
* Notes:
*       BorlandC's optimization is terrible!    When we put the code inline,
*       it generates fairly good code in the *following* segment (not in the Xor256
*       code itself).    If the call is made, the code following the call is awful!
*       The penalty is nearly 50%!    So we take the code size hit for inlining for
*       Borland, while Microsoft happily works with a call.
*
-*****************************************************************************/
#if defined(__BORLANDC__)       /* do it inline */
#define Xor32(dst,src,i) { ((DWORD *)dst)[i] = ((DWORD *)src)[i] ^ tmpX; }
#define  Xor256(dst,src,b)                                      ¥
        {                                                                               ¥
        register DWORD tmpX=0x01010101u * b;¥
        for (i=0;i<64;i+=4)                                             ¥
```

```
                { Xor32(dst,src,i   ); Xor32(dst,src,i+1); Xor32(dst,src,i+2); Xor32(dst,src,i+3); }
        ¥
        }
#else                                                   /* do it as a function call */
void Xor256(void *dst,void *src,BYTE b)
        {
        register DWORD x=b*0x01010101u;         /* replicate byte to all four bytes */
        register DWORD *d=(DWORD *)dst;
        register DWORD *s=(DWORD *)src;
#define X_8(N)    { d[N]=s[N] ^ x; d[N+1]=s[N+1] ^ x; }
#define X_32(N)  { X_8(N); X_8(N+2); X_8(N+4); X_8(N+6); }
        X_32(0 ); X_32( 8); X_32(16); X_32(24);         /* all inline */
        d+=32;   /* keep offsets small! */
        s+=32;
        X_32(0 ); X_32( 8); X_32(16); X_32(24);         /* all inline */
        }
#endif

/*
+*****************************************************************************
*
* Function Name:        reKey
*
* Function:                     Initialize the Twofish key schedule from key32
*
* Arguments:            key                     =       ptr to keyInstance to be initialized
*
* Return:                       TRUE on success
*
* Notes:
*       Here we precompute all the round subkeys, although that is not actually
*       required.   For example, on a smartcard, the round subkeys can
*       be generated on-the-fly     using f32()
*
-*****************************************************************************/
int reKey(keyInstance *key)
        {
        int             i,j,k64Cnt,keyLen;
        int             subkeyCnt;
        DWORDA=0,B=0,q;
        DWORD
        sKey[MAX_KEY_BITS/64],k32e[MAX_KEY_BITS/64],k32o[MAX_KEY_BITS/64];
        BYTE    L0[256],L1[256]; /* small local 8-bit permutations */

#if VALIDATE_PARMS
  #if ALIGN32
        if (((int)key) & 3)
                return BAD_ALIGN32;
        if ((key->keyLen % 64) || (key->keyLen < MIN_KEY_BITS))
                return BAD_KEY_INSTANCE;
  #endif
#endif

        if (needToBuildMDS)                     /* do this one time only */
                BuildMDS();

#define  F32(res,x,k32)    ¥
        {
                                                ¥
```

```
        DWORD t=x;
                                        ¥
        switch (k64Cnt & 3)
                                    ¥
            {
                                        ¥
                case 0:   /* same as 4 */
                ¥
                                        b0(t)     = p8(04)[b0(t)] ^ b0(k32[3]);
        ¥
                                        b1(t)     = p8(14)[b1(t)] ^ b1(k32[3]);
        ¥
                                        b2(t)     = p8(24)[b2(t)] ^ b2(k32[3]);
        ¥
                                        b3(t)     = p8(34)[b3(t)] ^ b3(k32[3]);
        ¥
                                    /* fall thru, having pre-processed t */                ¥
                case 3:             b0(t)     = p8(03)[b0(t)] ^ b0(k32[2]);                ¥
                                        b1(t)     = p8(13)[b1(t)] ^ b1(k32[2]);
        ¥
                                        b2(t)     = p8(23)[b2(t)] ^ b2(k32[2]);
        ¥
                                        b3(t)     = p8(33)[b3(t)] ^ b3(k32[2]);
        ¥
                                    /* fall thru, having pre-processed t */                ¥
                case 2:    /* 128-bit keys (optimize for this case) */    ¥
                           res=     MDStab[0][p8(01)[p8(02)[b0(t)] ^ b0(k32[1])] ^ b0(k32[0])] ^
        ¥
                                        MDStab[1][p8(11)[p8(12)[b1(t)]    ^    b1(k32[1])]    ^
b1(k32[0])] ^       ¥
                                        MDStab[2][p8(21)[p8(22)[b2(t)]    ^    b2(k32[1])]    ^
b2(k32[0])] ^       ¥
                                        MDStab[3][p8(31)[p8(32)[b3(t)]    ^    b3(k32[1])]    ^
b3(k32[0])] ;       ¥
                }
                                            ¥
        }


#if !CHECK_TABLE
#if defined(USE_ASM)                        /* only do this if not using assember */
if (!(useAsm & 4))
#endif
#endif
        {
        subkeyCnt = ROUND_SUBKEYS + 2*key->numRounds;
        keyLen=key->keyLen;
        k64Cnt=(keyLen+63)/64;              /* number of 64-bit key words */
        for (i=0,j=k64Cnt-1;i<k64Cnt;i++,j--)
                {                                                          /*    split    into
even/odd key dwords */
                k32e[i]=key->key32[2*i   ];
                k32o[i]=key->key32[2*i+1];
                /* compute S-box keys using (12,8) Reed-Solomon code over GF(256) */
                sKey[j]=key->sboxKeys[j]=RS_MDS_Encode(k32e[i],k32o[i]); /* reverse order */
                }
        }

#ifdef USE_ASM
```

```
if (useAsm & 4)
        {
        #if defined(COMPILE_KEY) && defined(USE_ASM)
                key->keySig             = VALID_SIG;                    /* show that we
are initialized */
                key->codeSize   = sizeof(key->compiledCode);    /* set size */
        #endif
        reKey_86(key);
        }
else
#endif
        {
        for (i=q=0;i<subkeyCnt/2;i++,q+=SK_STEP)
                {                                                       /* compute round
subkeys for PHT */
                F32(A,q         ,k32e);         /* A uses even key dwords */
                F32(B,q+SK_BUMP,k32o);                  /* B uses odd  key dwords */
                B = ROL(B,8);
                key->subKeys[2*i  ] = A+B;      /* combine with a PHT */
                B = A + 2*B;
                key->subKeys[2*i+1] = ROL(B,SK_ROTL);
                }
#if !defined(ZERO_KEY)
        switch (keyLen)  /* case out key length for speed in generating S-boxes */
                {
                case 128:
                #if defined(FULL_KEY) || defined(PART_KEY)
#if BIG_TAB
                        #define  one128(N,J)    sbSet(N,i,J,L0[i+J])
                        #define  sb128(N) {
        ¥
                                BYTE *qq=bigTab[N][b##N(sKey[1])];      ¥
                                Xor256(L0,qq,b##N(sKey[0]));            ¥
                                for (i=0;i<256;i+=2) { one128(N,0); one128(N,1); } }
#else
                        #define  one128(N,J)    sbSet(N,i,J,p8(N##1)[L0[i+J]]^k0)
                        #define  sb128(N) {                                     ¥
                                Xor256(L0,p8(N##2),b##N(sKey[1]));      ¥
                                { register DWORD k0=b##N(sKey[0]);      ¥
                                for (i=0;i<256;i+=2) { one128(N,0); one128(N,1); } } }
#endif
                #elif defined(MIN_KEY)
                        #define  sb128(N) Xor256(_sBox8_(N),p8(N##2),b##N(sKey[1]))
                #endif
                        sb128(0); sb128(1); sb128(2); sb128(3);
                        break;
                case 192:
                #if defined(FULL_KEY) || defined(PART_KEY)
                        #define one192(N,J) sbSet(N,i,J,p8(N##1)[p8(N##2)[L0[i+J]]^k1]^k0)
                        #define  sb192(N) {
        ¥
                                Xor256(L0,p8(N##3),b##N(sKey[2]));      ¥
                                { register DWORD k0=b##N(sKey[0]);      ¥
                                  register DWORD k1=b##N(sKey[1]);      ¥
                                  for (i=0;i<256;i+=2) { one192(N,0); one192(N,1); } } }
                #elif defined(MIN_KEY)
                        #define one192(N,J) sbSet(N,i,J,p8(N##2)[L0[i+J]]^k1)
                        #define  sb192(N) {
        ¥
```

```
                                        Xor256(L0,p8(N##3),b##N(sKey[2]));          ¥
                                        { register DWORD k1=b##N(sKey[1]);          ¥
                                          for (i=0;i<256;i+=2) { one192(N,0); one192(N,1); } } }
                #endif
                        sb192(0); sb192(1); sb192(2); sb192(3);
                        break;
                case 256:
                #if defined(FULL_KEY) || defined(PART_KEY)
                        #define one256(N,J) sbSet(N,i,J,p8(N##1)[p8(N##2)[L0[i+J]]^k1]^k0)
                        #define  sb256(N) {
                                                ¥
                                        Xor256(L1,p8(N##4),b##N(sKey[3]));
                        ¥
                                        for (i=0;i<256;i+=2) {L0[i   ]=p8(N##3)[L1[i]];
        ¥

L0[i+1]=p8(N##3)[L1[i+1]]; }            ¥
                                        Xor256(L0,L0,b##N(sKey[2]));
                                ¥
                                        { register DWORD k0=b##N(sKey[0]);
                                ¥
                                          register DWORD k1=b##N(sKey[1]);
                                ¥
                                          for (i=0;i<256;i+=2) { one256(N,0); one256(N,1); } } }
                #elif defined(MIN_KEY)
                        #define one256(N,J) sbSet(N,i,J,p8(N##2)[L0[i+J]]^k1)
                        #define  sb256(N) {
                                                ¥
                                        Xor256(L1,p8(N##4),b##N(sKey[3]));
                        ¥
                                        for (i=0;i<256;i+=2) {L0[i   ]=p8(N##3)[L1[i]];
        ¥

L0[i+1]=p8(N##3)[L1[i+1]]; }            ¥
                                        Xor256(L0,L0,b##N(sKey[2]));
                                ¥
                                        { register DWORD k1=b##N(sKey[1]);
                                ¥
                                          for (i=0;i<256;i+=2) { one256(N,0); one256(N,1); } } }
                #endif
                        sb256(0); sb256(1);         sb256(2); sb256(3);
                        break;
                }
#endif
        }

#if CHECK_TABLE                                                 /*  sanity  check      vs.
pedagogical code*/
        {
        GetSboxKey;
        for (i=0;i<subkeyCnt/2;i++)
                {
                A = f32(i*SK_STEP           ,k32e,keyLen);  /* A uses even key dwords */
                B = f32(i*SK_STEP+SK_BUMP,k32o,keyLen);         /* B uses odd   key dwords
*/
                B = ROL(B,8);
                assert(key->subKeys[2*i   ] == A+   B);
                assert(key->subKeys[2*i+1] == ROL(A+2*B,SK_ROTL));
                }
```

```
  #if !defined(ZERO_KEY)                        /* any S-boxes to check? */
        for (i=q=0;i<256;i++,q+=0x01010101)
            assert(f32(q,key->sboxKeys,keyLen) == Fe32_(q,0));
  #endif
        }
#endif /* CHECK_TABLE */

        DebugDumpKey(key);

        if (key->direction == DIR_ENCRYPT)
            ReverseRoundSubkeys(key,DIR_ENCRYPT);       /* reverse the round
subkey order */

        return TRUE;
        }
/*
+*****************************************************************************
*
* Function Name:        makeKey
*
* Function:             Initialize the Twofish key schedule
*
* Arguments:            key             =       ptr to keyInstance to be initialized
*                       direction       =       DIR_ENCRYPT or
DIR_DECRYPT
*                       keyLen          =       # bits of key text at
*keyMaterial
*                       keyMaterial     =       ptr to hex ASCII chars
representing key bits
*
* Return:               TRUE on success
*                       else error code (e.g., BAD_KEY_DIR)
*
* Notes: This parses the key bits from keyMaterial.   Zeroes out unused key bits
*
-*****************************************************************************/
int makeKey(keyInstance *key, BYTE direction, int keyLen,CONST char *keyMaterial)
        {
#if VALIDATE_PARMS                              /* first, sanity check on parameters */
        if (key == NULL)
            return BAD_KEY_INSTANCE;/* must have a keyInstance to initialize */
        if ((direction != DIR_ENCRYPT) && (direction != DIR_DECRYPT))
            return BAD_KEY_DIR;             /* must have valid direction */
        if ((keyLen > MAX_KEY_BITS) || (keyLen < 8) || (keyLen & 0x3F))
            return BAD_KEY_MAT;             /* length must be valid */
        key->keySig = VALID_SIG;            /* show that we are initialized */
  #if ALIGN32
        if ((((int)key) & 3) || (((int)key->key32) & 3))
            return BAD_ALIGN32;
  #endif
#endif

        key->direction    = direction;/* set our cipher direction */
        key->keyLen             = (keyLen+63) & ~63;            /* round up to multiple of
64 */
        key->numRounds          = numRounds[(keyLen-1)/64];
        memset(key->key32,0,sizeof(key->key32));  /* zero unused bits */
        key->keyMaterial[MAX_KEY_SIZE]=0;       /* terminate ASCII string */
```

```
    if ((keyMaterial == NULL) || (keyMaterial[0]==0))
        return TRUE;                    /* allow a "dummy" call */

    if (ParseHexDword(keyLen,keyMaterial,key->key32,key->keyMaterial))
        return BAD_KEY_MAT;

    return reKey(key);                  /* generate round subkeys */
    }


/*
+*****************************************************************************
*
* Function Name:        cipherInit
*
* Function:             Initialize the Twofish cipher in a given mode
*
* Arguments:            cipher  =   ptr to cipherInstance to be initialized
*                       mode    =   MODE_ECB, MODE_CBC, or MODE_CFB1
*                       IV      =   ptr to hex ASCII test representing IV bytes
*
* Return:               TRUE on success
*                       else error code (e.g., BAD_CIPHER_MODE)
*
-*****************************************************************************/
int cipherInit(cipherInstance *cipher, BYTE mode,CONST char *IV)
    {
    int i;
#if VALIDATE_PARMS                      /* first, sanity check on parameters */
    if (cipher == NULL)
        return BAD_PARAMS;              /* must have a cipherInstance to initialize */
    if ((mode != MODE_ECB) && (mode != MODE_CBC) && (mode != MODE_CFB1))
        return BAD_CIPHER_MODE;         /* must have valid cipher mode */
    cipher->cipherSig   =   VALID_SIG;
  #if ALIGN32
    if ((((int)cipher) & 3) || (((int)cipher->IV) & 3) || (((int)cipher->iv32) & 3))
        return BAD_ALIGN32;
  #endif
#endif

    if ((mode != MODE_ECB) && (IV))/* parse the IV */
        {
        if (ParseHexDword(BLOCK_SIZE,IV,cipher->iv32,NULL))
            return BAD_IV_MAT;
        for (i=0;i<BLOCK_SIZE/32;i++)   /* make byte-oriented copy for CFB1 */
            ((DWORD *)cipher->IV)[i] = Bswap(cipher->iv32[i]);
        }

    cipher->mode        =   mode;

    return TRUE;
    }

/*
+*****************************************************************************
*
* Function Name:        blockEncrypt
```

```
*
* Function:                     Encrypt block(s) of data using Twofish
*
* Arguments:    cipher      =   ptr to already initialized cipherInstance
*               key         =   ptr to already initialized keyInstance
*               input       =   ptr to data blocks to be encrypted
*               inputLen    =   # bits to encrypt (multiple of blockSize)
*               outBuffer   =   ptr to where to put encrypted blocks
*
* Return:                       # bits ciphered (>= 0)
*                               else error code (e.g., BAD_CIPHER_STATE, BAD_KEY_MATERIAL)
*
* Notes: The only supported block size for ECB/CBC modes is BLOCK_SIZE bits.
*                   If inputLen is not a multiple of BLOCK_SIZE bits in those modes,
*                   an error BAD_INPUT_LEN is returned.  In CFB1 mode, all block
*                   sizes can be supported.
*
-*****************************************************************************/
int blockEncrypt(cipherInstance *cipher, keyInstance *key,CONST BYTE *input,
                                    int inputLen, BYTE *outBuffer)
        {
        int     i,n;                                            /* loop counters */
        DWORD x[BLOCK_SIZE/32];                         /* block being encrypted */
        DWORD t0,t1;                                    /* temp variables */
        int             rounds=key->numRounds;  /* number of rounds */
        BYTE    bit,bit0,ctBit,carry;           /* temps for CFB */

        /* make local copies of things for faster access */
        int             mode = cipher->mode;
        DWORD sk[TOTAL_SUBKEYS];
        DWORD IV[BLOCK_SIZE/32];

        GetSboxKey;

#if VALIDATE_PARMS
        if ((cipher == NULL) || (cipher->cipherSig != VALID_SIG))
                return BAD_CIPHER_STATE;
        if ((key == NULL) || (key->keySig != VALID_SIG))
                return BAD_KEY_INSTANCE;
        if ((rounds < 2) || (rounds > MAX_ROUNDS) || (rounds&1))
                return BAD_KEY_INSTANCE;
        if ((mode != MODE_CFB1) && (inputLen % BLOCK_SIZE))
                return BAD_INPUT_LEN;
  #if ALIGN32
        if ( (((int)cipher) & 3) || (((int)key          ) & 3) ||
                 (((int)input ) & 3) || (((int)outBuffer) & 3))
                return BAD_ALIGN32;
  #endif
#endif

        if (mode == MODE_CFB1)
                {       /* use recursion here to handle CFB, one block at a time */
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
```

```
                        {
                        blockEncrypt(cipher,key,cipher->IV,BLOCK_SIZE,(BYTE *)x);
                        bit0    = 0x80 >> (n & 7);/* which bit position in byte */
                        ctBit = (input[n/8] & bit0) ^ ((((BYTE *) x)[0] & 0x80) >> (n&7));
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) | ctBit;
                        carry = ctBit >> (7 - (n&7));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;       /* restore mode for next time */
                return inputLen;
                }

        /* here for ECB, CBC modes */
        if (key->direction != DIR_ENCRYPT)
                ReverseRoundSubkeys(key,DIR_ENCRYPT);           /*  reverse  the  round
subkey order */

#ifdef USE_ASM
        if ((useAsm & 1) && (inputLen))
  #ifdef COMPILE_KEY
                if (key->keySig == VALID_SIG)
                        return                          ((CipherProc                    *)(key-
>encryptFuncPtr))(cipher,key,input,inputLen,outBuffer);
  #else
                return (*blockEncrypt_86)(cipher,key,input,inputLen,outBuffer);
  #endif
#endif
        /* make local copy of subkeys for speed */
        memcpy(sk,key->subKeys,sizeof(DWORD)*(ROUND_SUBKEYS+2*rounds));
        if (mode == MODE_CBC)
                BlockCopy(IV,cipher->iv32)
        else
                IV[0]=IV[1]=IV[2]=IV[3]=0;

        for
(n=0;n<inputLen;n+=BLOCK_SIZE,input+=BLOCK_SIZE/8,outBuffer+=BLOCK_SIZE/8)
                {
#ifdef DEBUG
                DebugDump(input,"¥n",-1,0,0,0,1);
                if (cipher->mode == MODE_CBC)
                        DebugDump(cipher->iv32,"",IV_ROUND,0,0,0,0);
#endif
#define  LoadBlockE(N)     x[N]=Bswap(((DWORD   *)input)[N])   ^   sk[INPUT_WHITEN+N]   ^
IV[N]
                LoadBlockE(0);   LoadBlockE(1);   LoadBlockE(2);   LoadBlockE(3);
                DebugDump(x,"",0,0,0,0,0);
#define  EncryptRound(K,R,id)      ¥
                        t0              = Fe32##id(x[K    ],0);
        ¥
                        t1              = Fe32##id(x[K^1],3);
        ¥
                        x[K^3] = ROL(x[K^3],1);
                ¥
                        x[K^2]^= t0 +     t1 + sk[ROUND_SUBKEYS+2*(R)    ];         ¥
```

```
                    x[K^3]^= t0 + 2*t1 + sk[ROUND_SUBKEYS+2*(R)+1];     ¥
                    x[K^2] = ROR(x[K^2],1);
                ¥
                    DebugDump(x,"",rounds-(R),0,0,1,0);
#define         Encrypt2(R,id)  { EncryptRound(0,R+1,id); EncryptRound(2,R,id); }

#if defined(ZERO_KEY)
                switch (key->keyLen)
                    {
                    case 128:
                        for (i=rounds-2;i>=0;i-=2)
                            Encrypt2(i,_128);
                        break;
                    case 192:
                        for (i=rounds-2;i>=0;i-=2)
                            Encrypt2(i,_192);
                        break;
                    case 256:
                        for (i=rounds-2;i>=0;i-=2)
                            Encrypt2(i,_256);
                        break;
                    }
#else
                Encrypt2(14,_);
                Encrypt2(12,_);
                Encrypt2(10,_);
                Encrypt2( 8,_);
                Encrypt2( 6,_);
                Encrypt2( 4,_);
                Encrypt2( 2,_);
                Encrypt2( 0,_);
#endif

                /* need to do (or undo, depending on your point of view) final swap */
#if LittleEndian
#define StoreBlockE(N)  ((DWORD *)outBuffer)[N]=x[N^2] ^ sk[OUTPUT_WHITEN+N]
#else
#define StoreBlockE(N)  {   t0=x[N^2]   ^   sk[OUTPUT_WHITEN+N];    ((DWORD
*)outBuffer)[N]=Bswap(t0); }
#endif
                StoreBlockE(0);  StoreBlockE(1);  StoreBlockE(2);  StoreBlockE(3);
                if (mode == MODE_CBC)
                    {
                    IV[0]=Bswap(((DWORD *)outBuffer)[0]);
                    IV[1]=Bswap(((DWORD *)outBuffer)[1]);
                    IV[2]=Bswap(((DWORD *)outBuffer)[2]);
                    IV[3]=Bswap(((DWORD *)outBuffer)[3]);
                    }
#ifdef DEBUG
                DebugDump(outBuffer,"",rounds+1,0,0,0,1);
                if (cipher->mode == MODE_CBC)
                    DebugDump(cipher->iv32,"",IV_ROUND,0,0,0,0);
#endif
                }

        if (mode == MODE_CBC)
                BlockCopy(cipher->iv32,IV);

        return inputLen;
```

```
    }

/*
+*****************************************************************************
*
* Function Name:        blockDecrypt
*
* Function:                     Decrypt block(s) of data using Twofish
*
* Arguments:            cipher          =       ptr to already initialized cipherInstance
*                                       key                     =       ptr to already
initialized keyInstance
*                                       input           =       ptr to data blocks to be
decrypted
*                                       inputLen        =       # bits to encrypt (multiple
of blockSize)
*                                       outBuffer       =       ptr to where to put
decrypted blocks
*
* Return:                       # bits ciphered (>= 0)
*                                       else error code (e.g., BAD_CIPHER_STATE,
BAD_KEY_MATERIAL)
*
* Notes: The only supported block size for ECB/CBC modes is BLOCK_SIZE bits.
*               If inputLen is not a multiple of BLOCK_SIZE bits in those modes,
*               an error BAD_INPUT_LEN is returned.  In CFB1 mode, all block
*               sizes can be supported.
*
-*****************************************************************************/
int blockDecrypt(cipherInstance *cipher, keyInstance *key,CONST BYTE *input,
                                int inputLen, BYTE *outBuffer)
    {
    int     i,n;                                                    /* loop counters */
    DWORD x[BLOCK_SIZE/32];                         /* block being encrypted */
    DWORD t0,t1;                                            /* temp variables */
    int         rounds=key->numRounds;      /* number of rounds */
    BYTE    bit,bit0,ctBit,carry;           /* temps for CFB */

    /* make local copies of things for faster access */
    int         mode = cipher->mode;
    DWORD sk[TOTAL_SUBKEYS];
    DWORD IV[BLOCK_SIZE/32];

    GetSboxKey;

#if VALIDATE_PARMS
    if ((cipher == NULL) || (cipher->cipherSig != VALID_SIG))
            return BAD_CIPHER_STATE;
    if ((key == NULL) || (key->keySig != VALID_SIG))
            return BAD_KEY_INSTANCE;
    if ((rounds < 2) || (rounds > MAX_ROUNDS) || (rounds&1))
            return BAD_KEY_INSTANCE;
    if ((cipher->mode != MODE_CFB1) && (inputLen % BLOCK_SIZE))
            return BAD_INPUT_LEN;
  #if ALIGN32
    if ( (((int)cipher) & 3) || (((int)key      ) & 3) ||
             (((int)input)  & 3) || (((int)outBuffer) & 3))
            return BAD_ALIGN32;
  #endif
```

```
#endif

        if (cipher->mode == MODE_CFB1)
                {       /* use blockEncrypt here to handle CFB, one block at a time */
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,cipher->IV,BLOCK_SIZE,(BYTE *)x);
                        bit0    = 0x80 >> (n & 7);
                        ctBit = input[n/8] & bit0;
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) |
                                                        (ctBit ^ ((((BYTE *) x)[0] & 0x80)
>> (n&7)));
                        carry = ctBit >> (7 - (n&7));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;       /* restore mode for next time */
                return inputLen;
                }

        /* here for ECB, CBC modes */
        if (key->direction != DIR_DECRYPT)
                ReverseRoundSubkeys(key,DIR_DECRYPT);           /* reverse the round
subkey order */
#ifdef USE_ASM
        if ((useAsm & 2) && (inputLen))
  #ifdef COMPILE_KEY
                if (key->keySig == VALID_SIG)
                        return                          ((CipherProc                    *)(key-
>decryptFuncPtr))(cipher,key,input,inputLen,outBuffer);
  #else
                return (*blockDecrypt_86)(cipher,key,input,inputLen,outBuffer);
  #endif
#endif
        /* make local copy of subkeys for speed */
        memcpy(sk,key->subKeys,sizeof(DWORD)*(ROUND_SUBKEYS+2*rounds));
        if (mode == MODE_CBC)
                BlockCopy(IV,cipher->iv32)
        else
                IV[0]=IV[1]=IV[2]=IV[3]=0;

        for
(n=0;n<inputLen;n+=BLOCK_SIZE,input+=BLOCK_SIZE/8,outBuffer+=BLOCK_SIZE/8)
                {
                DebugDump(input,"¥n",rounds+1,0,0,0,1);
#define LoadBlockD(N) x[N^2]=Bswap(((DWORD *)input)[N]) ^ sk[OUTPUT_WHITEN+N]
                LoadBlockD(0);  LoadBlockD(1);  LoadBlockD(2);  LoadBlockD(3);

#define  DecryptRound(K,R,id)
        ¥
                        t0              = Fe32##id(x[K   ],0);
        ¥
                        t1              = Fe32##id(x[K^1],3);
        ¥
```

```
                        DebugDump(x,"",(R)+1,0,0,1,0);
        ¥
                        x[K^2] = ROL (x[K^2],1);
        ¥
                        x[K^2]^= t0 +     t1 + sk[ROUND_SUBKEYS+2*(R)   ];      ¥
                        x[K^3]^= t0 + 2*t1 + sk[ROUND_SUBKEYS+2*(R)+1];      ¥
                        x[K^3] = ROR (x[K^3],1);
        ¥

#define         Decrypt2(R,id)    { DecryptRound(2,R+1,id); DecryptRound(0,R,id); }

#if defined(ZERO_KEY)
                switch (key->keyLen)
                        {
                        case 128:
                                for (i=rounds-2;i>=0;i-=2)
                                        Decrypt2(i,_128);
                                break;
                        case 192:
                                for (i=rounds-2;i>=0;i-=2)
                                        Decrypt2(i,_192);
                                break;
                        case 256:
                                for (i=rounds-2;i>=0;i-=2)
                                        Decrypt2(i,_256);
                                break;
                        }
#else
                {
                Decrypt2(14,_);
                Decrypt2(12,_);
                Decrypt2(10,_);
                Decrypt2( 8,_);
                Decrypt2( 6,_);
                Decrypt2( 4,_);
                Decrypt2( 2,_);
                Decrypt2( 0,_);
                }
#endif
                DebugDump(x,"",0,0,0,0,0);
                if (cipher->mode == MODE_ECB)
                        {
#if LittleEndian
#define  StoreBlockD(N)  ((DWORD *)outBuffer)[N] = x[N] ^ sk[INPUT_WHITEN+N]
#else
#define  StoreBlockD(N)  {  t0=x[N]^sk[INPUT_WHITEN+N];  ((DWORD  *)outBuffer)[N]  =
Bswap(t0); }
#endif
                        StoreBlockD(0);  StoreBlockD(1);  StoreBlockD(2);  StoreBlockD(3);
#undef   StoreBlockD
                        DebugDump(outBuffer,"",-1,0,0,0,1);
                        continue;
                        }
                else
                        {
#define  StoreBlockD(N)  x[N]     ^= sk[INPUT_WHITEN+N] ^ IV[N];          ¥
                                                IV[N]      =  Bswap(((DWORD  *)input)[N]);
        ¥
                                                ((DWORD *)outBuffer)[N] = Bswap(x[N]);
```

```
                                StoreBlockD(0);  StoreBlockD(1);  StoreBlockD(2);  StoreBlockD(3);
#undef  StoreBlockD
                                DebugDump(outBuffer,"",-1,0,0,0,1);
                                }
                        }
                if (mode == MODE_CBC)  /* restore iv32 to cipher */
                        BlockCopy(cipher->iv32,IV)

                return inputLen;
                }

#ifdef GetCodeSize
DWORD TwofishCodeSize(void)
                {
                DWORD x= Here(0);
#ifdef USE_ASM
                if (useAsm & 3)
                        return TwofishAsmCodeSize();
#endif
                return x - TwofishCodeStart();
                };
#endif


////////////////////////////////////////////

/* aes.h */

/* ---------- See examples at end of this file for typical usage -------- */

/* AES Cipher header file for ANSI C Submissions
        Lawrence E. Bassham III
        Computer Security Division
        National Institute of Standards and Technology

        This sample is to assist implementers developing to the
Cryptographic API Profile for AES Candidate Algorithm Submissions.
Please consult this document as a cross-reference.

        ANY CHANGES, WHERE APPROPRIATE, TO INFORMATION PROVIDED IN THIS
FILE
MUST BE DOCUMENTED. CHANGES ARE ONLY APPROPRIATE WHERE SPECIFIED
WITH
THE STRING "CHANGE POSSIBLE". FUNCTION CALLS AND THEIR PARAMETERS
CANNOT BE CHANGED. STRUCTURES CAN BE ALTERED TO ALLOW IMPLEMENTERS
TO
INCLUDE IMPLEMENTATION SPECIFIC INFORMATION.
*/

/* Includes:
        Standard include files
*/

#include<stdio.h>
#include"platform.h"                        /* platform-specific defines */

/*      Defines:
                Add any additional defines you need
*/
```

```
#define  DIR_ENCRYPT  0                  /* Are we encrpyting? */
#define  DIR_DECRYPT  1                  /* Are we decrpyting? */
#define  MODE_ECB             1                  /* Are we ciphering in ECB mode? */
#define  MODE_CBC             2                  /* Are we ciphering in CBC mode? */
#define  MODE_CFB1            3                  /* Are we ciphering in 1-bit CFB mode? */

#define  TRUE                 1
#define  FALSE                0

#define  BAD_KEY_DIR          -1        /* Key direction is invalid (unknown value) */
#define  BAD_KEY_MAT                    -2        /* Key material not of correct length */
#define  BAD_KEY_INSTANCE     -3        /* Key passed is not valid */
#define  BAD_CIPHER_MODE      -4        /* Params struct passed to cipherInit invalid */
#define  BAD_CIPHER_STATE     -5        /* Cipher in wrong state (e.g., not initialized) */

/* CHANGE POSSIBLE: inclusion of algorithm specific defines */
/* TWOFISH specific definitions */
#define          MAX_KEY_SIZE          64        /* # of ASCII chars needed to represent a key
*/
#define          MAX_IV_SIZE                     16        /* # of bytes needed to represent an
IV */
#define          BAD_INPUT_LEN                   -6        /* inputLen not a multiple of block
size */
#define          BAD_PARAMS                      -7        /* invalid parameters */
#define          BAD_IV_MAT                      -8        /* invalid IV text */
#define          BAD_ENDIAN                      -9        /* incorrect endianness define */
#define          BAD_ALIGN32                     -10       /* incorrect 32-bit alignment */

#define          BLOCK_SIZE                      128       /* number of bits per block */
#define          MAX_ROUNDS                       16       /* max # rounds (for allocating
subkey array) */
#define          ROUNDS_128                       16       /* default number of rounds for
128-bit keys*/
#define          ROUNDS_192                       16       /* default number of rounds for
192-bit keys*/
#define          ROUNDS_256                       16       /* default number of rounds for
256-bit keys*/
#define          MAX_KEY_BITS          256       /* max number of bits of key */
#define          MIN_KEY_BITS          128       /* min number of bits of key (zero pad) */
#define          VALID_SIG        0x48534946     /* initialization signature ('FISH') */
#define          MCT_OUTER                       400       /* MCT outer loop */
#define          MCT_INNER                  10000 /* MCT inner loop */
#define          REENTRANT                        1        /* nonzero forces reentrant code
(slightly slower) */

#define          INPUT_WHITEN                    0         /* subkey array indices */
#define          OUTPUT_WHITEN                   ( INPUT_WHITEN + BLOCK_SIZE/32)
#define          ROUND_SUBKEYS                   (OUTPUT_WHITEN   +   BLOCK_SIZE/32)
          /* use 2 * (# rounds) */
#define          TOTAL_SUBKEYS                   (ROUND_SUBKEYS + 2*MAX_ROUNDS)

/* Typedefs:
          Typedef'ed data storage elements. Add any algorithm specific
          parameters at the bottom of the structs as appropriate.
*/

typedef unsigned char BYTE;
typedef  unsigned long DWORD;             /* 32-bit unsigned quantity */
```

```
typedef DWORD fullSbox[4][256];

/* The structure for key information */
typedef struct
        {
        BYTE direction;                                 /* Key used for encrypting or
decrypting? */
#if ALIGN32
        BYTE dummyAlign[3];                             /* keep 32-bit alignment */
#endif
        int  keyLen;                                    /* Length of the key */
        char keyMaterial[MAX_KEY_SIZE+4];/* Raw key data in ASCII */

        /* Twofish-specific parameters: */
        DWORD keySig;                                   /* set to VALID_SIG by makeKey()
*/
        int         numRounds;                          /* number of rounds in cipher */
        DWORD key32[MAX_KEY_BITS/32];       /* actual key bits, in dwords */
        DWORD sboxKeys[MAX_KEY_BITS/64];/* key bits used for S-boxes */
        DWORD subKeys[TOTAL_SUBKEYS];       /* round subkeys, input/output whitening
bits */
#if REENTRANT
        fullSbox sBox8x32;                              /* fully expanded S-box */
  #if defined(COMPILE_KEY) && defined(USE_ASM)
#undef  VALID_SIG
#define VALID_SIG       0x504D4F43              /* 'COMP':     C is compiled with -
DCOMPILE_KEY */
        DWORD cSig1;                                    /* set after first "compile" (zero at
"init") */
        void *encryptFuncPtr;                   /* ptr to asm encrypt function */
        void *decryptFuncPtr;                   /* ptr to asm decrypt function */
        DWORD codeSize;                                 /* size of compiledCode */
        DWORD cSig2;                                    /* set after first "compile" */
        BYTE  compiledCode[5000];               /* make room for the code itself */
  #endif
#endif
        } keyInstance;

/* The structure for cipher information */
typedef struct
        {
        BYTE  mode;                                     /*          MODE_ECB,
MODE_CBC, or MODE_CFB1 */
#if ALIGN32
        BYTE dummyAlign[3];                             /* keep 32-bit alignment */
#endif
        BYTE  IV[MAX_IV_SIZE];                          /* CFB1 iv bytes  (CBC uses iv32)
*/

        /* Twofish-specific parameters: */
        DWORD cipherSig;                                /* set to VALID_SIG by cipherInit()
*/
        DWORD iv32[BLOCK_SIZE/32];              /* CBC IV bytes arranged as dwords */
        } cipherInstance;

/* Function protoypes */
int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial);

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);
```

```
int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                                        int inputLen, BYTE *outBuffer);

int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                                        int inputLen, BYTE *outBuffer);

int     reKey(keyInstance *key); /* do key schedule using modified key.keyDwords */

/* API to check table usage, for use in ECB_TBL KAT */
#define         TAB_DISABLE                     0
#define         TAB_ENABLE                      1
#define         TAB_RESET                       2
#define         TAB_QUERY                       3
#define         TAB_MIN_QUERY                   50
int TableOp(int op);


#define         CONST                           /* helpful C++ syntax sugar, NOP for ANSI
C */

#if BLOCK_SIZE == 128                   /* optimize block copies */
#define         Copy1(d,s,N)    ((DWORD *)(d))[N] = ((DWORD *)(s))[N]
#define         BlockCopy(d,s)  { Copy1(d,s,0);Copy1(d,s,1);Copy1(d,s,2);Copy1(d,s,3); }
#else
#define         BlockCopy(d,s)  { memcpy(d,s,BLOCK_SIZE/8); }
#endif


#ifdef TEST_2FISH
/*                                              ----- EXAMPLES -----

Unfortunately, the AES API is somewhat clumsy, and it is not entirely
obvious how to use the above functions.  In particular, note that
makeKey() takes an ASCII hex nibble key string (e.g., 32 characters
for a 128-bit key), which is rarely the way that keys are internally
represented.  The reKey() function uses instead the keyInstance.key32
array of key bits and is the preferred method.  In fact, makeKey()
initializes some internal keyInstance state, then parse the ASCII
string into the binary key32, and calls reKey().  To initialize the
keyInstance state, use a 'dummy' call to makeKey(); i.e., set the
keyMaterial parameter to NULL.  Then use reKey() for all key changes.
Similarly, cipherInit takes an IV string in ASCII hex, so a dummy setup
call with a null IV string will skip the ASCII parse.

Note that CFB mode is not well tested nor defined by AES, so using the
Twofish MODE_CFB it not recommended.  If you wish to implement a CFB mode,
build it external to the Twofish code, using the Twofish functions only
in ECB mode.

Below is a sample piece of code showing how the code is typically used
to set up a key, encrypt, and decrypt.  Error checking is somewhat limited
in this example.  Pseudorandom bytes are used for all key and text.

If you compile TWOFISH2.C or TWOFISH.C as a DOS (or Windows Console) app
with this code enabled, the test will be run.  For example, using
Borland C, you would compile using:
   BCC32 -DTEST_2FISH twofish2.c
to run the test on the optimized code, or
```

```
   BCC32 -DTEST_2FISH twofish.c
to run the test on the pedagogical code.

*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#define MAX_BLK_CNT             4               /* max # blocks per call in TestTwofish */
int TestTwofish(int mode,int keySize) /* keySize must be 128, 192, or 256 */
        {                                                       /* return 0 iff test passes
*/
        keyInstance     ki;                     /* key information, including tables */
        cipherInstance ci;                      /* keeps mode (ECB, CBC) and IV */
        BYTE    plainText[MAX_BLK_CNT*(BLOCK_SIZE/8)];
        BYTE cipherText[MAX_BLK_CNT*(BLOCK_SIZE/8)];
        BYTE decryptOut[MAX_BLK_CNT*(BLOCK_SIZE/8)];
        BYTE iv[BLOCK_SIZE/8];
        int    i,byteCnt;

        if (makeKey(&ki,DIR_ENCRYPT,keySize,NULL) != TRUE)
                return 1;                               /* 'dummy' setup for a 128-bit key
*/
        if (cipherInit(&ci,mode,NULL) != TRUE)
                return 1;                               /* 'dummy' setup for cipher */

        for (i=0;i<keySize/32;i++)  /* select key bits */
                ki.key32[i]=0x10003 * rand();
        reKey(&ki);                                     /* run the key schedule */

        if (mode != MODE_ECB)           /* set up random iv (if needed)*/
                {
                for (i=0;i<sizeof(iv);i++)
                        iv[i]=(BYTE) rand();
                memcpy(ci.iv32,iv,sizeof(ci.iv32));   /* copy the IV to ci */
                }

        /* select number of bytes to encrypt (multiple of block) */
        /* e.g., byteCnt = 16, 32, 48, 64 */
        byteCnt = (BLOCK_SIZE/8) * (1 + (rand() % MAX_BLK_CNT));

        for (i=0;i<byteCnt;i++)                 /* generate test data */
                plainText[i]=(BYTE) rand();

        /* encrypt the bytes */
        if (blockEncrypt(&ci,&ki, plainText,byteCnt*8,cipherText) != byteCnt*8)
                return 1;

        /* decrypt the bytes */
        if (mode != MODE_ECB)           /* first re-init the IV (if needed) */
                memcpy(ci.iv32,iv,sizeof(ci.iv32));

        if (blockDecrypt(&ci,&ki,cipherText,byteCnt*8,decryptOut) != byteCnt*8)
                return 1;

        /* make sure the decrypt output matches original plaintext */
        if (memcmp(plainText,decryptOut,byteCnt))
```

```
                return 1;

        return 0;                                               /* tests passed! */
        }

void main(void)
        {
        int testCnt,keySize;

        srand((unsigned) time(NULL));      /* randomize */

        for (keySize=128;keySize<=256;keySize+=64)
                for (testCnt=0;testCnt<10;testCnt++)
                        {
                        if (TestTwofish(MODE_ECB,keySize))
                                { printf("ECB Failure at keySize=%d",keySize); return; }
                        if (TestTwofish(MODE_CBC,keySize))
                                { printf("CBC Failure at keySize=%d",keySize); return; }
                        }
        printf("Tests passed");
        }
#endif /* TEST_2FISH */


////////////////////////////////////////////////////////////////////

#ifdef DEBUG    /* keep these macros common so they are same for both versions */
CONST int debugCompile=         1;
extern  int     debug;
extern  void DebugIO(CONST char *s);      /* display the debug output */

#define DebugDump(x,s,R,XOR,doRot,showT,needBswap)      ¥
        { if (debug) _Dump(x,s,R,XOR,doRot,showT,needBswap,t0,t1); }
#define  DebugDumpKey(key) { if (debug) _DumpKey(key); }
#define  IV_ROUND       -100

void _Dump(CONST void *p,CONST char *s,int R,int XOR,int doRot,int showT,int needBswap,
                DWORD t0,DWORD t1)
        {
        char line[512];    /* build output here */
        int   i,n;
        DWORD q[4];

        if (R == IV_ROUND)
                sprintf(line,"%sIV:      ",s);
        else
                sprintf(line,"%sR[%2d]: ",s,R);
        for (n=0;line[n];n++) ;

        for (i=0;i<4;i++)
                {
                q[i]=((CONST DWORD *)p)[i^(XOR)];
                if (needBswap) q[i]=Bswap(q[i]);
                }

        sprintf(line+n,"x= %08lX   %08lX   %08lX   %08lX.",
                        ROR(q[0],doRot*(R   )/2),
                        ROL(q[1],doRot*(R   )/2),
                        ROR(q[2],doRot*(R+1)/2),
```

```
                        ROL(q[3],doRot*(R+1)/2));
        for (;line[n];n++) ;

        if (showT)
                sprintf(line+n,"       t0=%08lX. t1=%08lX.",t0,t1);
        for (;line[n];n++) ;

        sprintf(line+n,"¥n");
        DebugIO(line);
        }

void _DumpKey(CONST keyInstance *key)
        {
        char    line[512];
        int             i;
        int             k64Cnt=(key->keyLen+63)/64;     /* round up to next multiple of 64
bits */
        int             subkeyCnt = ROUND_SUBKEYS + 2*key->numRounds;

        sprintf(line,";¥n;makeKey:    Input key               -->   S-box key       [%s]¥n",
                (key->direction == DIR_ENCRYPT) ? "Encrypt" : "Decrypt");
        DebugIO(line);
        for (i=0;i<k64Cnt;i++)      /* display in RS format */
                {
                sprintf(line,";%12s %08lX %08lX   -->   %08lX¥n","",
                        key->key32[2*i+1],key->key32[2*i],key->sboxKeys[k64Cnt-1-i]);
                DebugIO(line);
                }
        sprintf(line,";%11sSubkeys¥n","");
        DebugIO(line);
        for (i=0;i<subkeyCnt/2;i++)
                {
                sprintf(line,";%12s          %08lX          %08lX%s¥n","",key->subKeys[2*i],key-
>subKeys[2*i+1],
                        (2*i ==   INPUT_WHITEN) ? "    Input whiten" :
                        (2*i == OUTPUT_WHITEN) ? "   Output whiten" :
                    (2*i == ROUND_SUBKEYS) ? "   Round subkeys" : "");
                DebugIO(line);
                }
        DebugIO(";¥n");
        }
#else
CONST int debugCompile=         0;
#define DebugDump(x,s,R,XOR,doRot,showT,needBswap)
#define  DebugDumpKey(key)
#endif


////////////////////////////////////////////////////

/*****************************************************************************
        PLATFORM.H    -- Platform-specific defines for TWOFISH code

        Submitters:
                Bruce Schneier, Counterpane Systems
                Doug Whiting,    Hi/fn
                John Kelsey,     Counterpane Systems
                Chris Hall,                 Counterpane Systems
                David Wagner,    UC Berkeley
```

Code Author: Doug Whiting, Hi/fn

Version 1.00 April 1998

Copyright 1998, Hi/fn and Counterpane Systems. All rights reserved.

Notes:
* Tab size is set to 4 characters in this file

**********/

```
/* use intrinsic rotate if possible */
#define  ROL(x,n) (((x) << ((n) & 0x1F)) | ((x) >> (32-((n) & 0x1F))))
#define  ROR(x,n) (((x) >> ((n) & 0x1F)) | ((x) << (32-((n) & 0x1F))))

#if (0) && defined(__BORLANDC__) && (__BORLANDC__ >= 0x462)
#error "!!!This does not work for some reason!!!"
#include<stdlib.h>                                  /* get prototype for _lrotl() , _lrotr() */
#pragma inline __lrotl__
#pragma inline __lrotr__
#undef  ROL                                         /* get rid of inefficient definitions */
#undef  ROR
#define  ROL(x,n)       __lrotl__(x,n)              /* use compiler intrinsic rotations */
#define  ROR(x,n)       __lrotr__(x,n)
#endif

#ifdef _MSC_VER
#include<stdlib.h>                                  /* get prototypes for rotation functions */
#undef  ROL
#undef  ROR
#pragma intrinsic(_lrotl,_lrotr)                    /* use intrinsic compiler rotations */
#define  ROL(x,n)       _lrotl(x,n)
#define  ROR(x,n)       _lrotr(x,n)
#endif

#ifndef _M_IX86
#ifdef  __BORLANDC__
#define  _M_IX86                                    300             /* make sure this is defined for Intel CPUs */
#endif
#endif

#ifdef _M_IX86
#define         LittleEndian            1           /* e.g., 1 for Pentium, 0 for 68K */
#define         ALIGN32                 0           /* need dword alignment? (no for Pentium) */
#else   /* non-Intel platforms */
#define         LittleEndian            0           /* (assume big endian */
#define         ALIGN32                 1           /* (assume need alignment for non-Intel) */
#endif

#if LittleEndian
#define         Bswap(x)                (x)         /* NOP for little-endian machines */
```

```
#define		ADDR_XOR			0		/* NOP for little-endian machines */
#else
#define		Bswap(x)			((ROR(x,8) & 0xFF00FF00) | (ROL(x,8) & 0x00FF00FF))
#define		ADDR_XOR			3		/* convert byte address in dword */
#endif

/*		Macros for extracting bytes from dwords (correct for endianness) */
#define	_b(x,N)	(((BYTE *)&x)[((N) & 3) ^ ADDR_XOR]) /* pick bytes out of a dword */

#define		b0(x)			_b(x,0)		/* extract LSB of DWORD */
#define		b1(x)			_b(x,1)
#define		b2(x)			_b(x,2)
#define		b3(x)			_b(x,3)		/* extract MSB of DWORD */


//////////////////////////////////////////////////////

// stdafx.h : 標準のシステム インクルード ファイル、
//		または参照回数が多く、かつあまり変更されない
//		プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__FDA4C46D_8426_4948_A0C0_F265D6A12C31__INCLUDED_)
#define AFX_STDAFX_H__FDA4C46D_8426_4948_A0C0_F265D6A12C31__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif
// !defined(AFX_STDAFX_H__FDA4C46D_8426_4948_A0C0_F265D6A12C31__INCLUDED_)


//////////////////////////////////////////////////////////////

/*****************************************************************************
	TABLE.H		-- Tables, macros, constants for Twofish S-boxes and MDS matrix

	Submitters:
		Bruce Schneier, Counterpane Systems
		Doug Whiting,	Hi/fn
		John Kelsey,	Counterpane Systems
		Chris Hall,		Counterpane Systems
		David Wagner,	UC Berkeley

	Code Author:		Doug Whiting,	Hi/fn
```

```
        Version  1.00                April 1998

        Copyright 1998, Hi/fn and Counterpane Systems.  All rights reserved.

        Notes:
                *       Tab size is set to 4 characters in this file
                *       These definitions should be used in optimized and unoptimized
                        versions to insure consistency.

*****************************************************************************/

/* for computing subkeys */
#define  SK_STEP                        0x02020202u
#define  SK_BUMP                        0x01010101u
#define  SK_ROTL                        9

/* Reed-Solomon code parameters: (12,8) reversible code
        g(x) = x**4 + (a + 1/a) x**3 + a x**2 + (a + 1/a) x + 1
    where a = primitive root of field generator 0x14D */
#define  RS_GF_FDBK             0x14D           /* field generator */
#define  RS_rem(x)              ¥
        { BYTE  b  = (BYTE) (x >> 24);
                                                ¥
          DWORD g2 = ((b << 1) ^ ((b & 0x80) ? RS_GF_FDBK : 0 )) & 0xFF;          ¥
          DWORD g3 = ((b >> 1) & 0x7F) ^ ((b & 1) ? RS_GF_FDBK >> 1 : 0 ) ^ g2 ; ¥
          x = (x << 8) ^ (g3 << 24) ^ (g2 << 16) ^ (g3 << 8) ^ b;
         ¥
        }

/*      Macros for the MDS matrix
*       The MDS matrix is (using primitive polynomial 169):
*       01  EF  5B  5B
*       5B  EF  EF  01
*       EF  5B  01  EF
*       EF  01  EF  5B
*-----------------------------------------------------------------
* More statistical properties of this matrix (from MDS.EXE output):
*
* Min Hamming weight (one byte difference) =  8. Max=26.  Total =  1020.
* Prob[8]:      7     23     42     20     52     95     88     94    121    128     91
*              102     76     41     24      8      4      1      3      0      0      0
* Runs[8]:      2      4      5      6      7      8      9     11
* MSBs[8]:      1      4     15      8     18     38     40     43
* HW= 8: 05040705 0A080E0A 14101C14 28203828 50407050 01499101 A080E0A0
* HW= 9: 04050707 080A0E0E 10141C1C 20283838 40507070 80A0E0E0 C6432020 07070504
*        0E0E0A08 1C1C1410 38382820 70705040 E0E0A080 202043C6 05070407 0A0E080E
*        141C101C 28382038 50704070 A0E080E0 4320C620 02924B02 089A4508
* Min Hamming weight (two byte difference) =  3. Max=28.  Total = 390150.
* Prob[3]:      7     18     55    149    270    914   2185   5761 11363 20719 32079
*            43492 51612 53851 52098 42015 31117 20854 11538   6223   2492   1033
* MDS OK, ROR:   6+   7+   8+   9+ 10+ 11+ 12+ 13+ 14+ 15+ 16+
*                  17+ 18+ 19+ 20+ 21+ 22+ 23+ 24+ 25+ 26+
*/
#define  MDS_GF_FDBK            0x169   /* primitive polynomial for GF(256)*/
#define  LFSR1(x) ( ((x) >> 1)  ^ (((x) & 0x01) ?    MDS_GF_FDBK/2 : 0))
#define  LFSR2(x) ( ((x) >> 2)  ^ (((x) & 0x02) ?    MDS_GF_FDBK/2 : 0)  ¥
                                                                ^     (((x)     &     0x01)     ?
MDS_GF_FDBK/4 : 0))
```

```
#define  Mx_1(x) ((DWORD)   (x))             /* force result to dword so << will work */
#define  Mx_X(x) ((DWORD) ((x) ^                LFSR2(x)))    /* 5B */
#define  Mx_Y(x) ((DWORD) ((x) ^ LFSR1(x) ^ LFSR2(x)))    /* EF */

#define  M00              Mul_1
#define  M01              Mul_Y
#define  M02              Mul_X
#define  M03              Mul_X

#define  M10              Mul_X
#define  M11              Mul_Y
#define  M12              Mul_Y
#define  M13              Mul_1

#define  M20              Mul_Y
#define  M21              Mul_X
#define  M22              Mul_1
#define  M23              Mul_Y

#define  M30              Mul_Y
#define  M31              Mul_1
#define  M32              Mul_Y
#define  M33              Mul_X

#define  Mul_1   Mx_1
#define  Mul_X   Mx_X
#define  Mul_Y   Mx_Y

/*       Define the fixed p0/p1 permutations used in keyed S-box lookup.
         By changing the following constant definitions for P_ij, the S-boxes will
         automatically get changed in all the Twofish source code. Note that P_i0 is
         the "outermost" 8x8 permutation applied.   See the f32() function to see
         how these constants are to be    used.
*/
#define  P_00     1                                   /* "outermost" permutation */
#define  P_01     0
#define  P_02     0
#define  P_03     (P_01^1)                    /* "extend" to larger key sizes */
#define  P_04     1

#define  P_10     0
#define  P_11     0
#define  P_12     1
#define  P_13     (P_11^1)
#define  P_14     0

#define  P_20     1
#define  P_21     1
#define  P_22     0
#define  P_23     (P_21^1)
#define  P_24     0

#define  P_30     0
#define  P_31     1
#define  P_32     1
#define  P_33     (P_31^1)
#define  P_34     1
```

```
#define  p8(N)     P8x8[P_##N]                    /* some syntax shorthand */

/* fixed 8x8 permutation S-boxes */

/*****************************************************************************
*   07:07:14   05/30/98   [4x4]   TestCnt=256. keySize=128. CRC=4BD14D9E.
* maxKeyed:   dpMax = 18. lpMax =100. fixPt =   8. skXor =   0. skDup =   6.
* log2(dpMax[ 6..18])=     --- 15.42   1.33   0.89   4.05   7.98 12.05
* log2(lpMax[ 7..12])=   9.32   1.01   1.16   4.23   8.02 12.45
* log2(fixPt[ 0.. 8])=   1.44   1.44   2.44   4.06   6.01   8.21 11.07 14.09 17.00
* log2(skXor[ 0.. 0])
* log2(skDup[ 0.. 6])=     ---   2.37   0.44   3.94   8.36 13.04 17.99
*****************************************************************************/
CONST BYTE P8x8[2][256]=
        {
/*   p0:    */
/*   dpMax        = 10.   lpMax        = 64.   cycleCnt=    1   1   1   0.            */
/* 817D6F320B59ECA4.ECB81235F4A6709D.BA5E6D90C8F32471.D7F4126E9B3085CA. */
/* Karnaugh maps:
*   0111 0001 0011 1010. 0001 1001 1100 1111. 1001 1110 0011 1110. 1101 0101 1111 1001.
*   0101 1111 1100 0100. 1011 0101 0010 0000. 0101 1000 1100 0101. 1000 0111 0011 0010.
*   0000 1001 1110 1101. 1011 1000 1010 0011. 0011 1001 0101 0000. 0100 0010 0101 1011.
*   0111 0100 0001 0110. 1000 1011 1110 1001. 0011 0011 1001 1101. 1101 0101 0000 1100.
*/
        {
        0xA9, 0x67, 0xB3, 0xE8, 0x04, 0xFD, 0xA3, 0x76,
        0x9A, 0x92, 0x80, 0x78, 0xE4, 0xDD, 0xD1, 0x38,
        0x0D, 0xC6, 0x35, 0x98, 0x18, 0xF7, 0xEC, 0x6C,
        0x43, 0x75, 0x37, 0x26, 0xFA, 0x13, 0x94, 0x48,
        0xF2, 0xD0, 0x8B, 0x30, 0x84, 0x54, 0xDF, 0x23,
        0x19, 0x5B, 0x3D, 0x59, 0xF3, 0xAE, 0xA2, 0x82,
        0x63, 0x01, 0x83, 0x2E, 0xD9, 0x51, 0x9B, 0x7C,
        0xA6, 0xEB, 0xA5, 0xBE, 0x16, 0x0C, 0xE3, 0x61,
        0xC0, 0x8C, 0x3A, 0xF5, 0x73, 0x2C, 0x25, 0x0B,
        0xBB, 0x4E, 0x89, 0x6B, 0x53, 0x6A, 0xB4, 0xF1,
        0xE1, 0xE6, 0xBD, 0x45, 0xE2, 0xF4, 0xB6, 0x66,
        0xCC, 0x95, 0x03, 0x56, 0xD4, 0x1C, 0x1E, 0xD7,
        0xFB, 0xC3, 0x8E, 0xB5, 0xE9, 0xCF, 0xBF, 0xBA,
        0xEA, 0x77, 0x39, 0xAF, 0x33, 0xC9, 0x62, 0x71,
        0x81, 0x79, 0x09, 0xAD, 0x24, 0xCD, 0xF9, 0xD8,
        0xE5, 0xC5, 0xB9, 0x4D, 0x44, 0x08, 0x86, 0xE7,
        0xA1, 0x1D, 0xAA, 0xED, 0x06, 0x70, 0xB2, 0xD2,
        0x41, 0x7B, 0xA0, 0x11, 0x31, 0xC2, 0x27, 0x90,
        0x20, 0xF6, 0x60, 0xFF, 0x96, 0x5C, 0xB1, 0xAB,
        0x9E, 0x9C, 0x52, 0x1B, 0x5F, 0x93, 0x0A, 0xEF,
        0x91, 0x85, 0x49, 0xEE, 0x2D, 0x4F, 0x8F, 0x3B,
        0x47, 0x87, 0x6D, 0x46, 0xD6, 0x3E, 0x69, 0x64,
        0x2A, 0xCE, 0xCB, 0x2F, 0xFC, 0x97, 0x05, 0x7A,
        0xAC, 0x7F, 0xD5, 0x1A, 0x4B, 0x0E, 0xA7, 0x5A,
        0x28, 0x14, 0x3F, 0x29, 0x88, 0x3C, 0x4C, 0x02,
        0xB8, 0xDA, 0xB0, 0x17, 0x55, 0x1F, 0x8A, 0x7D,
        0x57, 0xC7, 0x8D, 0x74, 0xB7, 0xC4, 0x9F, 0x72,
        0x7E, 0x15, 0x22, 0x12, 0x58, 0x07, 0x99, 0x34,
        0x6E, 0x50, 0xDE, 0x68, 0x65, 0xBC, 0xDB, 0xF8,
        0xC8, 0xA8, 0x2B, 0x40, 0xDC, 0xFE, 0x32, 0xA4,
        0xCA, 0x10, 0x21, 0xF0, 0xD3, 0x5D, 0x0F, 0x00,
        0x6F, 0x9D, 0x36, 0x42, 0x4A, 0x5E, 0xC1, 0xE0
        },
/*   p1:    */
```

```
/*   dpMax        = 10.   lpMax        = 64.   cycleCnt=    2   0   0   1.            */
/* 28BDF76E31940AC5.1E2B4C376DA5F908.4C75169A0ED82B3F.B951C3DE647F208A. */
/* Karnaugh maps:
*   0011 1001 0010 0111. 1010 0111 0100 0110. 0011 0001 1111 0100. 1111 1000 0001 1100.
*   1100 1111 1111 1010. 0011 0011 1110 0100. 1001 0110 0100 0011. 0101 0110 1011 1011.
*   0010 0100 0011 0101. 1100 1000 1000 1110. 0111 1111 0010 0110. 0000 1010 0000 0011.
*   1101 1000 0010 0001. 0110 1001 1110 0101. 0001 0100 0101 0111. 0011 1011 1111 0010.
*/
        {
        0x75, 0xF3, 0xC6, 0xF4, 0xDB, 0x7B, 0xFB, 0xC8,
        0x4A, 0xD3, 0xE6, 0x6B, 0x45, 0x7D, 0xE8, 0x4B,
        0xD6, 0x32, 0xD8, 0xFD, 0x37, 0x71, 0xF1, 0xE1,
        0x30, 0x0F, 0xF8, 0x1B, 0x87, 0xFA, 0x06, 0x3F,
        0x5E, 0xBA, 0xAE, 0x5B, 0x8A, 0x00, 0xBC, 0x9D,
        0x6D, 0xC1, 0xB1, 0x0E, 0x80, 0x5D, 0xD2, 0xD5,
        0xA0, 0x84, 0x07, 0x14, 0xB5, 0x90, 0x2C, 0xA3,
        0xB2, 0x73, 0x4C, 0x54, 0x92, 0x74, 0x36, 0x51,
        0x38, 0xB0, 0xBD, 0x5A, 0xFC, 0x60, 0x62, 0x96,
        0x6C, 0x42, 0xF7, 0x10, 0x7C, 0x28, 0x27, 0x8C,
        0x13, 0x95, 0x9C, 0xC7, 0x24, 0x46, 0x3B, 0x70,
        0xCA, 0xE3, 0x85, 0xCB, 0x11, 0xD0, 0x93, 0xB8,
        0xA6, 0x83, 0x20, 0xFF, 0x9F, 0x77, 0xC3, 0xCC,
        0x03, 0x6F, 0x08, 0xBF, 0x40, 0xE7, 0x2B, 0xE2,
        0x79, 0x0C, 0xAA, 0x82, 0x41, 0x3A, 0xEA, 0xB9,
        0xE4, 0x9A, 0xA4, 0x97, 0x7E, 0xDA, 0x7A, 0x17,
        0x66, 0x94, 0xA1, 0x1D, 0x3D, 0xF0, 0xDE, 0xB3,
        0x0B, 0x72, 0xA7, 0x1C, 0xEF, 0xD1, 0x53, 0x3E,
        0x8F, 0x33, 0x26, 0x5F, 0xEC, 0x76, 0x2A, 0x49,
        0x81, 0x88, 0xEE, 0x21, 0xC4, 0x1A, 0xEB, 0xD9,
        0xC5, 0x39, 0x99, 0xCD, 0xAD, 0x31, 0x8B, 0x01,
        0x18, 0x23, 0xDD, 0x1F, 0x4E, 0x2D, 0xF9, 0x48,
        0x4F, 0xF2, 0x65, 0x8E, 0x78, 0x5C, 0x58, 0x19,
        0x8D, 0xE5, 0x98, 0x57, 0x67, 0x7F, 0x05, 0x64,
        0xAF, 0x63, 0xB6, 0xFE, 0xF5, 0xB7, 0x3C, 0xA5,
        0xCE, 0xE9, 0x68, 0x44, 0xE0, 0x4D, 0x43, 0x69,
        0x29, 0x2E, 0xAC, 0x15, 0x59, 0xA8, 0x0A, 0x9E,
        0x6E, 0x47, 0xDF, 0x34, 0x35, 0x6A, 0xCF, 0xDC,
        0x22, 0xC9, 0xC0, 0x9B, 0x89, 0xD4, 0xED, 0xAB,
        0x12, 0xA2, 0x0D, 0x52, 0xBB, 0x02, 0x2F, 0xA9,
        0xD7, 0x61, 0x1E, 0xB4, 0x50, 0x04, 0xF6, 0xC2,
        0x16, 0x25, 0x86, 0x56, 0x55, 0x09, 0xBE, 0x91
        }
        };
```

１０．３．３　鍵の形式

鍵は次のような 3 行からなるテキストファイルです。

```
128
128
40B6E45ED3FE4CA2D1E14EDD9C48FC68
```

1 行目は、鍵の長さ
2 行目は、ブロックサイズ
3 行目は鍵を 16 進数で表したものです。

１０．４　Serpent 暗号ソースコード

インターネット上で公開されている Serpent 暗号のソースコードを利用しましたが、本来のソースコードには誤りがあると判断して一部を変更しています。本来のものと比較検討をして、自分で正しいと考える方を採用してください。私のホームページから本来のソースコードへのリンクが張ってあります。

> Serpent is free for use. No assurance of any kind is given for any
> modified version, and the security of such version may depend on the
> changes.

ということですが、修正は MODE_CFB1 に関する部分で、本来のソースコードを大幅に変更してあります。この部分は皆様が必ず、本来のものと比較検討をしてください。もちろんこの修正が必要であり、しかも正しいと私（宇山靖政）は確信しています。

１０．４．１　暗号化部分

```
/* Copyright (C) 1998 Ross Anderson, Eli Biham, Lars Knudsen
 * All rights reserved.
 *
 * This code is freely distributed for AES selection process.
 * No other use is allowed.
 *
 * Copyright remains of the copyright holders, and as such any Copyright
 * notices in the code are not to be removed.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted only for the AES selection process, provided
 * that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *      notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE
LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed without the authors permission.
 *   i.e. this code cannot simply be copied and put under another distribution
 * licence [including the GNU Public Licence.]
 */

#include "stdafx.h"
#include "serpent.h"
#include "serpentsboxes.h"
```

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#define BLOCK_SIZE 128
#define DWORD unsigned long

/*  The functions  */
int serpent_makeKey(keyInstance *key, BYTE direction, int keyLen,
                                char *keyMaterial)
{
   unsigned long i,j;
   unsigned long w[132],k[132];
   int rc;

   if(direction != DIR_ENCRYPT &&
       direction != DIR_DECRYPT)
      return BAD_KEY_DIR;

   if(keyLen>256 || keyLen<1)
      return BAD_KEY_MAT;

   key->direction=direction;
   key->keyLen=keyLen;
   strncpy(key->keyMaterial, keyMaterial, MAX_KEY_SIZE+1);

   rc=serpent_convert_from_string(keyLen, keyMaterial,key->key);
   if(rc<=0)
      return BAD_KEY_MAT;

   for(i=0; i<keyLen/32; i++)
      w[i]=key->key[i];
   if(keyLen<256)
      w[i]=(key->key[i]&((1L<<((keyLen&31)))-1))|(1L<<((keyLen&31)));
   for(i++; i<8; i++)
      w[i]=0;
   for(i=8; i<16; i++)
      w[i]=ROL(w[i-8]^w[i-5]^w[i-3]^w[i-1]^PHI^(i-8),11);
   for(i=0; i<8; i++)
      w[i]=w[i+8];
   for(i=8; i<132; i++)
      w[i]=ROL(w[i-8]^w[i-5]^w[i-3]^w[i-1]^PHI^i,11);

   RND03(w[  0], w[  1], w[  2], w[  3], k[  0], k[  1], k[  2], k[  3]);
   RND02(w[  4], w[  5], w[  6], w[  7], k[  4], k[  5], k[  6], k[  7]);
   RND01(w[  8], w[  9], w[ 10], w[ 11], k[  8], k[  9], k[ 10], k[ 11]);
   RND00(w[ 12], w[ 13], w[ 14], w[ 15], k[ 12], k[ 13], k[ 14], k[ 15]);
   RND31(w[ 16], w[ 17], w[ 18], w[ 19], k[ 16], k[ 17], k[ 18], k[ 19]);
   RND30(w[ 20], w[ 21], w[ 22], w[ 23], k[ 20], k[ 21], k[ 22], k[ 23]);
   RND29(w[ 24], w[ 25], w[ 26], w[ 27], k[ 24], k[ 25], k[ 26], k[ 27]);
   RND28(w[ 28], w[ 29], w[ 30], w[ 31], k[ 28], k[ 29], k[ 30], k[ 31]);
   RND27(w[ 32], w[ 33], w[ 34], w[ 35], k[ 32], k[ 33], k[ 34], k[ 35]);
   RND26(w[ 36], w[ 37], w[ 38], w[ 39], k[ 36], k[ 37], k[ 38], k[ 39]);
   RND25(w[ 40], w[ 41], w[ 42], w[ 43], k[ 40], k[ 41], k[ 42], k[ 43]);
   RND24(w[ 44], w[ 45], w[ 46], w[ 47], k[ 44], k[ 45], k[ 46], k[ 47]);
   RND23(w[ 48], w[ 49], w[ 50], w[ 51], k[ 48], k[ 49], k[ 50], k[ 51]);
```

```
    RND22(w[ 52], w[ 53], w[ 54], w[ 55], k[ 52], k[ 53], k[ 54], k[ 55]);
    RND21(w[ 56], w[ 57], w[ 58], w[ 59], k[ 56], k[ 57], k[ 58], k[ 59]);
    RND20(w[ 60], w[ 61], w[ 62], w[ 63], k[ 60], k[ 61], k[ 62], k[ 63]);
    RND19(w[ 64], w[ 65], w[ 66], w[ 67], k[ 64], k[ 65], k[ 66], k[ 67]);
    RND18(w[ 68], w[ 69], w[ 70], w[ 71], k[ 68], k[ 69], k[ 70], k[ 71]);
    RND17(w[ 72], w[ 73], w[ 74], w[ 75], k[ 72], k[ 73], k[ 74], k[ 75]);
    RND16(w[ 76], w[ 77], w[ 78], w[ 79], k[ 76], k[ 77], k[ 78], k[ 79]);
    RND15(w[ 80], w[ 81], w[ 82], w[ 83], k[ 80], k[ 81], k[ 82], k[ 83]);
    RND14(w[ 84], w[ 85], w[ 86], w[ 87], k[ 84], k[ 85], k[ 86], k[ 87]);
    RND13(w[ 88], w[ 89], w[ 90], w[ 91], k[ 88], k[ 89], k[ 90], k[ 91]);
    RND12(w[ 92], w[ 93], w[ 94], w[ 95], k[ 92], k[ 93], k[ 94], k[ 95]);
    RND11(w[ 96], w[ 97], w[ 98], w[ 99], k[ 96], k[ 97], k[ 98], k[ 99]);
    RND10(w[100], w[101], w[102], w[103], k[100], k[101], k[102], k[103]);
    RND09(w[104], w[105], w[106], w[107], k[104], k[105], k[106], k[107]);
    RND08(w[108], w[109], w[110], w[111], k[108], k[109], k[110], k[111]);
    RND07(w[112], w[113], w[114], w[115], k[112], k[113], k[114], k[115]);
    RND06(w[116], w[117], w[118], w[119], k[116], k[117], k[118], k[119]);
    RND05(w[120], w[121], w[122], w[123], k[120], k[121], k[122], k[123]);
    RND04(w[124], w[125], w[126], w[127], k[124], k[125], k[126], k[127]);
    RND03(w[128], w[129], w[130], w[131], k[128], k[129], k[130], k[131]);

    for(i=0; i<=32; i++)
        for(j=0; j<4; j++)
            key->subkeys[i][j] = k[4*i+j];

    return TRUE;
}

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV)
{
//    int i;
    int rc;

    if((mode != MODE_ECB) &&
       (mode != MODE_CBC) &&
       (mode != MODE_CFB1))
        return BAD_CIPHER_MODE;

    cipher->mode = mode;                /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
    cipher->blockSize=128;
    if(mode != MODE_ECB)
        {
            rc=serpent_convert_from_string(cipher->blockSize, IV, (unsigned long*)cipher->IV);
            if(rc<=0)
                return BAD_CIPHER_STATE;
        }

    return TRUE;
}

int blockEncrypt(cipherInstance *cipher,
                 keyInstance *key,
                 BYTE *input,
                 int inputLen,
                 BYTE *outBuffer)
{
    unsigned long t[4];
    int    b, n, i;
    DWORD x[BLOCK_SIZE/32];
```

```
    BYTE bit, bit0, ctBit, carry;

    /*
     * Note about optimization: the code becomes slower of the calls to
     * serpent_encrypt and serpent_decrypt are replaced by inlined code.
     * (tested on Pentium 133MMX)
     */

    switch(cipher->mode)
      {
      case MODE_ECB:
        for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
           serpent_encrypt((unsigned long*)input,(unsigned long*) outBuffer, key->subkeys);
        return inputLen;

      case MODE_CBC:
        t[0] = ((unsigned long*)cipher->IV)[0];
        t[1] = ((unsigned long*)cipher->IV)[1];
        t[2] = ((unsigned long*)cipher->IV)[2];
        t[3] = ((unsigned long*)cipher->IV)[3];
        for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
          {
            t[0] ^= ((unsigned long*)input)[0];
            t[1] ^= ((unsigned long*)input)[1];
            t[2] ^= ((unsigned long*)input)[2];
            t[3] ^= ((unsigned long*)input)[3];
            serpent_encrypt(t, t, key->subkeys);
            ((unsigned long*)outBuffer)[0] = t[0];
            ((unsigned long*)outBuffer)[1] = t[1];
            ((unsigned long*)outBuffer)[2] = t[2];
            ((unsigned long*)outBuffer)[3] = t[3];
          }
        ((unsigned long*)cipher->IV)[0] = t[0];
        ((unsigned long*)cipher->IV)[1] = t[1];
        ((unsigned long*)cipher->IV)[2] = t[2];
        ((unsigned long*)cipher->IV)[3] = t[3];

        return inputLen;

      case MODE_CFB1:
                        cipher->mode = MODE_ECB;        /* do encryption in ECB */
                        for (n=0;n<inputLen;n++)
                                {
                                blockEncrypt(cipher,key,(BYTE       *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                                bit0    = 0x80 >> (n & 7);/* which bit position in byte */
                                ctBit = (input[n/8] & bit0) ^ ((((BYTE *) x)[0] & 0x80) >> (n&7));
                                outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) | ctBit;
                                carry = ctBit >> (7 - (n&7));
                                for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                        {
                                        bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                        cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                        carry = bit;
                                        }
                                }
                        cipher->mode = MODE_CFB1;       /* restore mode for next time */
                        return inputLen;
```

```
    default:
      return BAD_CIPHER_STATE;
    }
}

int blockDecrypt(cipherInstance *cipher,
                 keyInstance *key,
                 BYTE *input,
                 int inputLen,
                 BYTE *outBuffer)
{
  unsigned long t[4];
  int   b, n, i;
  DWORD x[BLOCK_SIZE/32];
  BYTE bit, bit0, ctBit, carry;

  switch(cipher->mode)
    {
    case MODE_ECB:
      for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
          serpent_decrypt( (unsigned long*)input, (unsigned long*)outBuffer, key->subkeys);
      return inputLen;

    case MODE_CBC:
      t[0] = ((unsigned long*)cipher->IV)[0];
      t[1] = ((unsigned long*)cipher->IV)[1];
      t[2] = ((unsigned long*)cipher->IV)[2];
      t[3] = ((unsigned long*)cipher->IV)[3];
      for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
        {
          serpent_decrypt((unsigned long*)input, (unsigned long*)outBuffer,key->subkeys);
          ((unsigned long*)outBuffer)[0] ^= t[0];
          ((unsigned long*)outBuffer)[1] ^= t[1];
          ((unsigned long*)outBuffer)[2] ^= t[2];
          ((unsigned long*)outBuffer)[3] ^= t[3];
          t[0] = ((unsigned long*)input)[0];
          t[1] = ((unsigned long*)input)[1];
          t[2] = ((unsigned long*)input)[2];
          t[3] = ((unsigned long*)input)[3];
        }
      ((unsigned long*)cipher->IV)[0] = t[0];
      ((unsigned long*)cipher->IV)[1] = t[1];
      ((unsigned long*)cipher->IV)[2] = t[2];
      ((unsigned long*)cipher->IV)[3] = t[3];
      return inputLen;

    case MODE_CFB1://blockDecrypt
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,(BYTE       *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                        bit0    = 0x80 >> (n & 7);
                        ctBit = input[n/8] & bit0;
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) |
                                                        (ctBit ^ ((((BYTE *) x)[0] & 0x80)
>> (n&7)));
                        carry = ctBit >> (7 - (n&7));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
```

```
                                        {
                                        bit = cipher->IV[i] >> 7;     /* save next "carry" from shift */
                                        cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                        carry = bit;
                                        }
                                }
                        cipher->mode = MODE_CFB1;       /* restore mode for next time */
                        return inputLen;

        default:
           return BAD_CIPHER_STATE;
        }
}

void serpent_encrypt(unsigned long plaintext[4],
                        unsigned long ciphertext[4],
                        unsigned long subkeys[33][4])
{
   register unsigned long x0, x1, x2, x3;
   register unsigned long y0, y1, y2, y3;

   x0=plaintext[0];
   x1=plaintext[1];
   x2=plaintext[2];
   x3=plaintext[3];

   /* Start to encrypt the plaintext x */
   keying(x0, x1, x2, x3, subkeys[ 0]);
   RND00(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 1]);
   RND01(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 2]);
   RND02(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 3]);
   RND03(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 4]);
   RND04(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 5]);
   RND05(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 6]);
   RND06(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 7]);
   RND07(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 8]);
   RND08(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[ 9]);
   RND09(x0, x1, x2, x3, y0, y1, y2, y3);
   transform(y0, y1, y2, y3, x0, x1, x2, x3);
   keying(x0, x1, x2, x3, subkeys[10]);
   RND10(x0, x1, x2, x3, y0, y1, y2, y3);
```

```
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[11]);
RND11(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[12]);
RND12(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[13]);
RND13(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[14]);
RND14(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[15]);
RND15(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[16]);
RND16(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[17]);
RND17(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[18]);
RND18(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[19]);
RND19(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[20]);
RND20(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[21]);
RND21(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[22]);
RND22(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[23]);
RND23(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[24]);
RND24(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[25]);
RND25(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[26]);
RND26(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[27]);
RND27(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[28]);
RND28(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[29]);
RND29(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[30]);
```

```
    RND30(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[31]);
    RND31(x0, x1, x2, x3, y0, y1, y2, y3);
    x0 = y0; x1 = y1; x2 = y2; x3 = y3;
    keying(x0, x1, x2, x3, subkeys[32]);
    /* The ciphertext is now in x */

    ciphertext[0] = x0;
    ciphertext[1] = x1;
    ciphertext[2] = x2;
    ciphertext[3] = x3;
}

void serpent_decrypt(unsigned long ciphertext[4],
                     unsigned long plaintext[4],
                     unsigned long subkeys[33][4])
{
    register unsigned long x0, x1, x2, x3;
    register unsigned long y0, y1, y2, y3;

    x0=ciphertext[0];
    x1=ciphertext[1];
    x2=ciphertext[2];
    x3=ciphertext[3];

    /* Start to decrypt the ciphertext x */
    keying(x0, x1, x2, x3, subkeys[32]);
    InvRND31(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[31]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND30(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[30]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND29(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[29]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND28(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[28]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND27(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[27]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND26(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[26]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND25(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[25]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND24(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[24]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND23(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[23]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND22(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[22]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND21(x0, x1, x2, x3, y0, y1, y2, y3);
```

```
keying(y0, y1, y2, y3, subkeys[21]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND20(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[20]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND19(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[19]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND18(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[18]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND17(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[17]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND16(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[16]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND15(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[15]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND14(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[14]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND13(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[13]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND12(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[12]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND11(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[11]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND10(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[10]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND09(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 9]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND08(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 8]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND07(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 7]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND06(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 6]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND05(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 5]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND04(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 4]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND03(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 3]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND02(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 2]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
```

```
    InvRND01(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 1]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND00(x0, x1, x2, x3, y0, y1, y2, y3);
    x0 = y0; x1 = y1; x2 = y2; x3 = y3;
    keying(x0, x1, x2, x3, subkeys[ 0]);
    /* The plaintext is now in x */

    plaintext[0] = x0;
    plaintext[1] = x1;
    plaintext[2] = x2;
    plaintext[3] = x3;
}

#define min(x,y) (((x)<(y))?(x):(y))

int serpent_convert_from_string(int len, char *str, unsigned long *val)
/* the size of val must be at least the next multiple of 32 */
/* bits after len bits */
{
    int is, iv;
    int slen=min((int)strlen(str), (len+3)/4);

    if(len<0)
        return -1;                  /* Error!!! */

    if(len>slen*4 || len<slen*4-3)
        return -1;                  /* Error!!! */

    for(is=0; is<slen; is++)
        if(((str[is]<'0')||(str[is]>'9')) &&
           ((str[is]<'A')||(str[is]>'F')) &&
           ((str[is]<'a')||(str[is]>'f')))
            return -1;  /* Error!!! */

    for(is=slen, iv=0; is>=8; is-=8, iv++)
        {
            unsigned long t;
            sscanf(&str[is-8], "%08lX", &t);
            val[iv] = t;
        }
    if(is>0)
        {
            char tmp[10];
            unsigned long t;
            strncpy(tmp, str, is);
            tmp[is] = 0;
            sscanf(tmp, "%08lX", &t);
            val[iv++] = t;
        }
    for(; iv<(len+31)/32; iv++)
        val[iv] = 0;
    return iv;
}

char *serpent_convert_to_string(int len, unsigned long val[8], char *str)
/* str must have at least (len+3)/4+1 bytes. */
{
    int i;
```

```
    if(len<0)
        return (char *)-1;                  /* Error!!! */

    str[0] = 0;
    i=len/32;
    if(len&31>0)
        {
            char tmp[10];
            sprintf(tmp, "%08lX", val[i]&(((len&31)<<1)-1));
            strcat(str, &tmp[8-(((len&31)+3)/4)]);
        }
    for(i--; i>=0; i--)
        {
            char tmp[10];
            sprintf(tmp, "%08lX", val[i]);
            strcat(str, tmp);
        }
    return str;
}

//////////////////////////////////////////////////////////////

// SerpentEC.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include <stdio.h>
#include <time.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <malloc.h>
#include "serpent.h"

#define file_len(x) (unsigned long)x

int s;
keyInstance keyI;
cipherInstance cipherI;


int main(int argc, char* argv[])
{
        FILE *fkey =0, *fin = 0, *fout = 0;
    fpos_t flen;
    unsigned long len, rlen, blen4;
        int mode,klen,blen,rc;
        char c_mode[4], c_klen[8], c_key[96], c_cini[64];
        unsigned char pbuf[2048], cbuf[2048];

        blen4 = 2048;

        printf( "Start!¥n" );

        if( argc != 4 ){                                    // 使い方の誤り
                printf( "argc != 4 ¥n" );
                return -1;
```

```
	}

	if((fkey = fopen(argv[1], "rt")) == NULL){
		printf("Can not find key file for encryption. ¥n");
		return (-1);
	}

	fgets( c_mode, 4, fkey );
	fgets( c_klen, 8, fkey );
	fgets( c_key, 96, fkey );
	fgets( c_cini, 64, fkey );

	mode = atoi(c_mode);
	klen = atoi(c_klen);
	blen = 128;

	if(klen<56 || 256<klen){
		printf("Wrong key size. ¥n");
		return (-1);
	}

	/*Set mode*/
	if(mode == 1){
		rc=cipherInit(&cipherI, MODE_ECB, "");
	}
	if(mode == 2){
		rc=cipherInit(&cipherI, MODE_CBC, c_cini);
	}
	if(mode == 3){
		rc=cipherInit(&cipherI, MODE_CFB1, c_cini);
	}
	if(rc<=0){
		printf("モード設定が出来ません。");
		return(-2);
	}

	serpent_makeKey(&keyI, DIR_ENCRYPT,    klen, c_key );

	if((fin = fopen(argv[2], "rb")) == NULL){
		printf("Can not open plane text file. ¥n");
		return (-1);
	}

fseek(fin, 0, SEEK_END);
fgetpos(fin, &flen);
rlen = file_len(flen);
// reset to start
fseek(fin, 0, SEEK_SET);

	if((fout = fopen(argv[3], "wb")) == NULL){
		printf("Can not open encrypted data file. ¥n");
		return (-1);
	}

	s = sizeof(unsigned int);
	// write the bytes of the file
	*((unsigned int*)pbuf) = rlen;

	len = (unsigned long) fread(pbuf+s, 1, blen4-s , fin);
```

```
	rlen -= len;
		rc=blockEncrypt(&cipherI, &keyI,(unsigned char*)(pbuf),8*blen4,(unsigned
char*)(cbuf));
	if(fwrite(cbuf, 1, blen4, fout) != (unsigned int)blen4)
		return -2;

	while(rlen > 0 && !feof(fin)){
		// read a block and reduce the remaining byte count
		len = (unsigned long)fread(pbuf, 1, blen4 , fin);
		rlen -= len;
			rc=blockEncrypt(&cipherI, &keyI,(unsigned char*)(pbuf),8*blen4,(unsigned
char*)(cbuf));
		if(fwrite(cbuf, 1, blen4, fout) != blen4){return -2;}
	}

		if(fkey){ fclose(fkey);	}
		if(fout){ fclose(fout);	}
		if(fin)	{ fclose(fin);	}
		printf( "End!¥n" );
		return 0;
}

//////////////////////////////////////////////////////////

// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//		SerpentEC.pch 生成されるプリコンパイル済ヘッダー
//		stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。

//////////////////////////////////////////////////////////

/*	aes.h	*/

/*	AES Cipher header file for ANSI C Submissions
	Lawrence E. Bassham III
	Computer Security Division
	National Institute of Standards and Technology

	April 15, 1998

	This sample is to assist implementers developing to the Cryptographic
API Profile for AES Candidate Algorithm Submissions.	Please consult this
document as a cross-reference.

	ANY CHANGES, WHERE APPROPRIATE, TO INFORMATION PROVIDED IN THIS FILE
MUST BE DOCUMENTED.	CHANGES ARE ONLY APPROPRIATE WHERE SPECIFIED
WITH
THE STRING "CHANGE POSSIBLE".	FUNCTION CALLS AND THEIR PARAMETERS
CANNOT
BE CHANGED.	STRUCTURES CAN BE ALTERED TO ALLOW IMPLEMENTERS TO
INCLUDE
IMPLEMENTATION SPECIFIC INFORMATION.
*/

/*	Includes:
```

```
        Standard include files
*/

#include <stdio.h>

/*  Defines:
        Add any additional defines you need
*/

#define     DIR_ENCRYPT     0     /*  Are we encrpyting?  */
#define     DIR_DECRYPT     1     /*  Are we decrpyting?  */
#define     MODE_ECB        1     /*  Are we ciphering in ECB mode?    */
#define     MODE_CBC        2     /*  Are we ciphering in CBC mode?    */
#define     MODE_CFB1       3     /*  Are we ciphering in 1-bit CFB mode? */
#define     TRUE            1
#define     FALSE           0

/*  Error Codes - CHANGE POSSIBLE: inclusion of additional error codes  */
#define     BAD_KEY_DIR         -1  /*  Key direction is invalid, e;g;,
                                        unknown value */
#define     BAD_KEY_MAT         -2  /*  Key material not of correct
                                        length */
#define     BAD_KEY_INSTANCE    -3  /*  Key passed is not valid  */
#define     BAD_CIPHER_MODE     -4  /*  Params struct passed to
                                        cipherInit invalid */
#define     BAD_CIPHER_STATE    -5  /*  Cipher in wrong state (e.g., not
                                        initialized) */

/*  CHANGE POSSIBLE:  inclusion of algorithm specific defines  */
#define     MAX_KEY_SIZE    64  /* # of ASCII char's needed to
                                        represent a key */
#define     MAX_IV_SIZE         32  /* # of ASCII char's needed to
                                        represent an IV  */

/*  Typedefs:

        Typedef'ed data storage elements.  Add any algorithm specific
parameters at the bottom of the structs as appropriate.
*/

typedef     unsigned char     BYTE;

/*  The structure for key information */
typedef struct {
        BYTE    direction;  /*  Key used for encrypting or decrypting? */
        int     keyLen;     /*  Length of the key  */
        char    keyMaterial[MAX_KEY_SIZE+1];  /*  Raw key data in ASCII, e.g.,
                                        what the user types or KAT values)*/
        /*  The following parameters are algorithm dependent, replace or
                add as necessary  */
        unsigned long key[8];               /* The key in binary */
        unsigned long subkeys[33][4];       /* Serpent subkeys */
        } keyInstance;

/*  The structure for cipher information */
typedef struct {
        BYTE    mode;           /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
        char    IV[MAX_IV_SIZE];  /* A possible Initialization Vector for
                                        ciphering */
```

```
        /*    Add any algorithm specific parameters needed here    */
        int     blockSize;              /* Sample: Handles non-128 bit block sizes
                                                (if available) */
        } cipherInstance;


/*    Function protoypes    */
int serpent_makeKey(keyInstance *key, BYTE direction, int keyLen,
                char *keyMaterial);

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                        int inputLen, BYTE *outBuffer);

int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                        int inputLen, BYTE *outBuffer);

int serpent_convert_from_string(int, char *, unsigned long *);
void serpent_encrypt(unsigned long [],unsigned long [], unsigned long [][4]);
void serpent_decrypt(unsigned long [],unsigned long [], unsigned long [][4]);

//////////////////////////////////////////////////////

/* Copyright (C) 1998 Ross Anderson, Eli Biham, Lars Knudsen
 * All rights reserved.
 *
 * This code is freely distributed for AES selection process.
 * No other use is allowed.
 *
 * Copyright remains of the copyright holders, and as such any Copyright
 * notices in the code are not to be removed.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted only for the AES selection process, provided
 * that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *      notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE
LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
```

```
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed without the authors permission.
 *    i.e. this code cannot simply be copied and put under another distribution
 * licence [including the GNU Public Licence.]
 */

/* S0:    3   8 15   1 10   6   5 11 14 13   4   2   7   0   9 12 */

/* depth = 5,7,4,2, Total gates=18 */
#define RND00(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t05, t06, t07, t08, t09, t11, t12, t13, t14, t15, t17, t01;¥
        t01 = b     ^ c   ; ¥
        t02 = a     | d   ; ¥
        t03 = a     ^ b   ; ¥
        z     = t02 ^ t01; ¥
        t05 = c     | z   ; ¥
        t06 = a     ^ d   ; ¥
        t07 = b     | c   ; ¥
        t08 = d     & t05; ¥
        t09 = t03 & t07; ¥
        y     = t09 ^ t08; ¥
        t11 = t09 & y   ; ¥
        t12 = c     ^ d   ; ¥
        t13 = t07 ^ t11; ¥
        t14 = b     & t06; ¥
        t15 = t06 ^ t13; ¥
        w     =       ~ t15; ¥
        t17 = w     ^ t14; ¥
        x     = t12 ^ t17; }

/* InvS0:   13   3 11   0 10   6   5 12   1 14   4   7 15   9   8   2 */

/* depth = 8,4,3,6, Total gates=19 */
#define InvRND00(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t12, t13, t14, t15, t17, t18,
t01;¥
        t01 = c     ^ d   ; ¥
        t02 = a     | b   ; ¥
        t03 = b     | c   ; ¥
        t04 = c     & t01; ¥
        t05 = t02 ^ t01; ¥
        t06 = a     | t04; ¥
        y     =       ~ t05; ¥
        t08 = b     ^ d   ; ¥
        t09 = t03 & t08; ¥
        t10 = d     | y   ; ¥
        x     = t09 ^ t06; ¥
        t12 = a     | t05; ¥
        t13 = x     ^ t12; ¥
        t14 = t03 ^ t10; ¥
        t15 = a     ^ c   ; ¥
        z     = t14 ^ t13; ¥
        t17 = t05 & t13; ¥
        t18 = t14 | t17; ¥
        w     = t15 ^ t18; }

/* S1:   15 12   2   7   9   0   5 10   1 11 14   8   6 13   3   4 */
```

```
/* depth = 10,7,3,5, Total gates=18 */
#define RND01(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t10, t11, t12, t13, t16, t17, t01;¥
        t01 = a     | d   ; ¥
        t02 = c     ^ d   ; ¥
        t03 =       ~ b   ; ¥
        t04 = a     ^ c   ; ¥
        t05 = a     | t03; ¥
        t06 = d     & t04; ¥
        t07 = t01 & t02; ¥
        t08 = b     | t06; ¥
        y   = t02 ^ t05; ¥
        t10 = t07 ^ t08; ¥
        t11 = t01 ^ t10; ¥
        t12 = y     ^ t11; ¥
        t13 = b     & d   ; ¥
        z   =       ~ t10; ¥
        x   = t13 ^ t12; ¥
        t16 = t10 | x   ; ¥
        t17 = t05 & t16; ¥
        w   = c     ^ t17; }

/* InvS1:   5   8   2 14 15   6 12   3 11   4   7   9   1 13 10   0 */

/* depth = 7,4,5,3, Total gates=18 */
#define InvRND01(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t14, t15, t17, t01;¥
        t01 = a     ^ b   ; ¥
        t02 = b     | d   ; ¥
        t03 = a     & c   ; ¥
        t04 = c     ^ t02; ¥
        t05 = a     | t04; ¥
        t06 = t01 & t05; ¥
        t07 = d     | t03; ¥
        t08 = b     ^ t06; ¥
        t09 = t07 ^ t06; ¥
        t10 = t04 | t03; ¥
        t11 = d     & t08; ¥
        y   =       ~ t09; ¥
        x   = t10 ^ t11; ¥
        t14 = a     | y   ; ¥
        t15 = t06 ^ x   ; ¥
        z   = t01 ^ t04; ¥
        t17 = c     ^ t15; ¥
        w   = t14 ^ t17; }

/* S2:   8   6   7   9   3 12 10 15 13   1 14   4   0 11   5   2 */

/* depth = 3,8,11,7, Total gates=16 */
#define RND02(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t05, t06, t07, t08, t09, t10, t12, t13, t14, t01;¥
        t01 = a     | c   ; ¥
        t02 = a     ^ b   ; ¥
        t03 = d     ^ t01; ¥
        w   = t02 ^ t03; ¥
        t05 = c     ^ w   ; ¥
        t06 = b     ^ t05; ¥
        t07 = b     | t05; ¥
```

```
        t08 = t01 & t06; ¥
        t09 = t03 ^ t07; ¥
        t10 = t02 | t09; ¥
        x   = t10 ^ t08; ¥
        t12 = a   | d  ; ¥
        t13 = t09 ^ x  ; ¥
        t14 = b   ^ t13; ¥
        z   =     ~ t09; ¥
        y   = t12 ^ t14; }

/* InvS2:  12  9 15  4 11 14  1  2  0  3  6 13  5  8 10  7 */

/* depth = 3,6,8,3, Total gates=18 */
#define InvRND02(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t06, t07, t08, t09, t10, t11, t12, t15, t16, t17, t01;¥
        t01 = a   ^ d  ; ¥
        t02 = c   ^ d  ; ¥
        t03 = a   & c  ; ¥
        t04 = b   | t02; ¥
        w   = t01 ^ t04; ¥
        t06 = a   | c  ; ¥
        t07 = d   | w  ; ¥
        t08 =     ~ d  ; ¥
        t09 = b   & t06; ¥
        t10 = t08 | t03; ¥
        t11 = b   & t07; ¥
        t12 = t06 & t02; ¥
        z   = t09 ^ t10; ¥
        x   = t12 ^ t11; ¥
        t15 = c   & z  ; ¥
        t16 = w   ^ x  ; ¥
        t17 = t10 ^ t15; ¥
        y   = t16 ^ t17; }

/* S3:   0 15 11  8 12  9  6  3 13  1  2  4 10  7  5 14 */

/* depth = 8,3,5,5, Total gates=18 */
#define RND03(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t13, t14, t15, t01;¥
        t01 = a   ^ c  ; ¥
        t02 = a   | d  ; ¥
        t03 = a   & d  ; ¥
        t04 = t01 & t02; ¥
        t05 = b   | t03; ¥
        t06 = a   & b  ; ¥
        t07 = d   ^ t04; ¥
        t08 = c   | t06; ¥
        t09 = b   ^ t07; ¥
        t10 = d   & t05; ¥
        t11 = t02 ^ t10; ¥
        z   = t08 ^ t09; ¥
        t13 = d   | z  ; ¥
        t14 = a   | t07; ¥
        t15 = b   & t13; ¥
        y   = t08 ^ t11; ¥
        w   = t14 ^ t15; ¥
        x   = t05 ^ t04; }

/* InvS3:  0  9 10  7 11 14  6 13  3  5 12  2  4  8 15  1 */
```

```
/* depth = 3,6,4,4, Total gates=17 */
#define InvRND03(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t09, t11, t12, t13, t14, t16, t01;¥
        t01 = c     | d   ; ¥
        t02 = a     | d   ; ¥
        t03 = c     ^ t02; ¥
        t04 = b     ^ t02; ¥
        t05 = a     ^ d   ; ¥
        t06 = t04 & t03; ¥
        t07 = b     & t01; ¥
        y     = t05 ^ t06; ¥
        t09 = a     ^ t03; ¥
        w     = t07 ^ t03; ¥
        t11 = w     | t05; ¥
        t12 = t09 & t11; ¥
        t13 = a     & y   ; ¥
        t14 = t01 ^ t05; ¥
        x     = b     ^ t12; ¥
        t16 = b     | t13; ¥
        z     = t14 ^ t16; }

/* S4:    1 15   8   3 12   0 11   6   2   5   4 10   9 14   7 13 */

/* depth = 6,7,5,3, Total gates=19 */
#define RND04(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t11, t12, t13, t14, t15, t16,
t01;¥
        t01 = a     | b   ; ¥
        t02 = b     | c   ; ¥
        t03 = a     ^ t02; ¥
        t04 = b     ^ d   ; ¥
        t05 = d     | t03; ¥
        t06 = d     & t01; ¥
        z     = t03 ^ t06; ¥
        t08 = z     & t04; ¥
        t09 = t04 & t05; ¥
        t10 = c     ^ t06; ¥
        t11 = b     & c   ; ¥
        t12 = t04 ^ t08; ¥
        t13 = t11 | t03; ¥
        t14 = t10 ^ t09; ¥
        t15 = a     & t05; ¥
        t16 = t11 | t12; ¥
        y     = t13 ^ t08; ¥
        x     = t15 ^ t16; ¥
        w     =       ~ t14; }

/* InvS4:    5   0   8   3 10   9   7 14   2 12 11   6   4 15 13   1 */

/* depth = 6,4,7,3, Total gates=17 */
#define InvRND04(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t09, t10, t11, t12, t13, t15, t01;¥
        t01 = b     | d   ; ¥
        t02 = c     | d   ; ¥
        t03 = a     & t01; ¥
        t04 = b     ^ t02; ¥
        t05 = c     ^ d   ; ¥
        t06 =       ~ t03; ¥
```

```
        t07 = a     & t04; ¥
        x   = t05 ^ t07; ¥
        t09 = x     | t06; ¥
        t10 = a     ^ t07; ¥
        t11 = t01 ^ t09; ¥
        t12 = d     ^ t04; ¥
        t13 = c     | t10; ¥
        z   = t03 ^ t12; ¥
        t15 = a     ^ t04; ¥
        y   = t11 ^ t13; ¥
        w   = t15 ^ t09; }

/* S5:  15  5  2 11  4 10  9 12  0  3 14  8 13  6  7  1 */

/* depth = 4,6,8,6, Total gates=17 */
#define RND05(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t14, t01;¥
        t01 = b     ^ d   ; ¥
        t02 = b     | d   ; ¥
        t03 = a     & t01; ¥
        t04 = c     ^ t02; ¥
        t05 = t03 ^ t04; ¥
        w   =       ~ t05; ¥
        t07 = a     ^ t01; ¥
        t08 = d     | w   ; ¥
        t09 = b     | t05; ¥
        t10 = d     ^ t08; ¥
        t11 = b     | t07; ¥
        t12 = t03 | w   ; ¥
        t13 = t07 | t10; ¥
        t14 = t01 ^ t11; ¥
        y   = t09 ^ t13; ¥
        x   = t07 ^ t08; ¥
        z   = t12 ^ t14; }

/* InvS5:   8 15  2  9  4  1 13 14 11  6  5  3  7 12 10  0 */

/* depth = 4,6,9,7, Total gates=17 */
#define InvRND05(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t12, t13, t15, t16, t01;¥
        t01 = a     & d   ; ¥
        t02 = c     ^ t01; ¥
        t03 = a     ^ d   ; ¥
        t04 = b     & t02; ¥
        t05 = a     & c   ; ¥
        w   = t03 ^ t04; ¥
        t07 = a     & w   ; ¥
        t08 = t01 ^ w   ; ¥
        t09 = b     | t05; ¥
        t10 =       ~ b   ; ¥
        x   = t08 ^ t09; ¥
        t12 = t10 | t07; ¥
        t13 = w     | x   ; ¥
        z   = t02 ^ t12; ¥
        t15 = t02 ^ t13; ¥
        t16 = b     ^ d   ; ¥
        y   = t16 ^ t15; }

/* S6:   7  2 12  5  8  4  6 11 14  9  1 15 13  3 10  0 */
```

```
/* depth = 8,3,6,3, Total gates=19 */
#define RND06(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t15, t17, t18,
t01;¥
        t01 = a     & d   ; ¥
        t02 = b     ^ c   ; ¥
        t03 = a     ^ d   ; ¥
        t04 = t01 ^ t02; ¥
        t05 = b     | c   ; ¥
        x     =       ~ t04; ¥
        t07 = t03 & t05; ¥
        t08 = b     & x   ; ¥
        t09 = a     | c   ; ¥
        t10 = t07 ^ t08; ¥
        t11 = b     | d   ; ¥
        t12 = c     ^ t11; ¥
        t13 = t09 ^ t10; ¥
        y     =       ~ t13; ¥
        t15 = x     & t03; ¥
        z     = t12 ^ t07; ¥
        t17 = a     ^ b   ; ¥
        t18 = y     ^ t15; ¥
        w     = t17 ^ t18; }

/* InvS6:   15 10   1 13   5   3   6   0   4   9 14   7   2 12   8 11 */

/* depth = 5,3,8,6, Total gates=19 */
#define InvRND06(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t12, t13, t14, t15, t16, t17,
t01;¥
        t01 = a     ^ c   ; ¥
        t02 =       ~ c   ; ¥
        t03 = b     & t01; ¥
        t04 = b     | t02; ¥
        t05 = d     | t03; ¥
        t06 = b     ^ d   ; ¥
        t07 = a     & t04; ¥
        t08 = a     | t02; ¥
        t09 = t07 ^ t05; ¥
        x     = t06 ^ t08; ¥
        w     =       ~ t09; ¥
        t12 = b     & w   ; ¥
        t13 = t01 & t05; ¥
        t14 = t01 ^ t12; ¥
        t15 = t07 ^ t13; ¥
        t16 = d     | t02; ¥
        t17 = a     ^ x   ; ¥
        z     = t17 ^ t15; ¥
        y     = t16 ^ t14; }

/* S7:    1 13 15   0 14   8   2 11   7   4 12 10   9   3   5   6 */

/* depth = 10,7,10,4, Total gates=19 */
#define RND07(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t11, t13, t14, t15, t16, t17,
t01;¥
        t01 = a     & c   ; ¥
        t02 =       ~ d   ; ¥
```

```
        t03 = a     & t02; ¥
        t04 = b     | t01; ¥
        t05 = a     & b   ; ¥
        t06 = c     ^ t04; ¥
        z   = t03 ^ t06; ¥
        t08 = c     | z   ; ¥
        t09 = d     | t05; ¥
        t10 = a     ^ t08; ¥
        t11 = t04 & z   ; ¥
        x   = t09 ^ t10; ¥
        t13 = b     ^ x   ; ¥
        t14 = t01 ^ x   ; ¥
        t15 = c     ^ t05; ¥
        t16 = t11 | t13; ¥
        t17 = t02 | t14; ¥
        w   = t15 ^ t17; ¥
        y   = a     ^ t16; }

/* InvS7:    3   0   6 13   9 14 15   8   5 12 11   7 10   1   4   2 */

/* depth = 9,7,3,3, Total gates=18 */
#define InvRND07(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t06, t07, t08, t09, t10, t11, t13, t14, t15, t16, t01;¥
        t01 = a     & b   ; ¥
        t02 = a     | b   ; ¥
        t03 = c     | t01; ¥
        t04 = d     & t02; ¥
        z   = t03 ^ t04; ¥
        t06 = b     ^ t04; ¥
        t07 = d     ^ z   ; ¥
        t08 =       ~ t07; ¥
        t09 = t06 | t08; ¥
        t10 = b     ^ d   ; ¥
        t11 = a     | d   ; ¥
        x   = a     ^ t09; ¥
        t13 = c     ^ t06; ¥
        t14 = c     & t11; ¥
        t15 = d     | x   ; ¥
        t16 = t01 | t10; ¥
        w   = t13 ^ t15; ¥
        y   = t14 ^ t16; }

#define RND08(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND09(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND10(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND11(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND12(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND13(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND14(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND15(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)
#define RND16(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND17(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND18(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND19(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND20(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND21(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND22(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND23(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)
#define RND24(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
```

```
#define RND25(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND26(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND27(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND28(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND29(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND30(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND31(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)

#define InvRND08(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND09(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND10(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND11(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND12(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND13(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND14(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND15(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)
#define InvRND16(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND17(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND18(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND19(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND20(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND21(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND22(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND23(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)
#define InvRND24(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND25(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND26(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND27(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND28(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND29(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND30(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND31(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)

/* Linear transformations and key mixing: */

#define ROL(x,n) ((((unsigned long)(x))<<(n)) | ¥
                        (((unsigned long)(x))>>(32-(n))))
#define ROR(x,n) ((((unsigned long)(x))<<(32-(n))) | ¥
                        (((unsigned long)(x))>>(n)))

#define transform(x0, x1, x2, x3, y0, y1, y2, y3) ¥
        y0 = ROL(x0, 13); ¥
        y2 = ROL(x2, 3); ¥
        y1 = x1 ^ y0 ^ y2; ¥
        y3 = x3 ^ y2 ^ ((unsigned long)y0)<<3; ¥
        y1 = ROL(y1, 1); ¥
        y3 = ROL(y3, 7); ¥
        y0 = y0 ^ y1 ^ y3; ¥
        y2 = y2 ^ y3 ^ ((unsigned long)y1<<7); ¥
        y0 = ROL(y0, 5); ¥
        y2 = ROL(y2, 22)

#define inv_transform(x0, x1, x2, x3, y0, y1, y2, y3) ¥
        y2 = ROR(x2, 22);¥
        y0 = ROR(x0, 5); ¥
        y2 = y2 ^ x3 ^ ((unsigned long)x1<<7); ¥
        y0 = y0 ^ x1 ^ x3; ¥
        y3 = ROR(x3, 7); ¥
        y1 = ROR(x1, 1); ¥
```

```
        y3 = y3 ^ y2 ^ ((unsigned long)y0)<<3; ¥
        y1 = y1 ^ y0 ^ y2; ¥
        y2 = ROR(y2, 3); ¥
        y0 = ROR(y0, 13)

#define keying(x0, x1, x2, x3, subkey) ¥
                                x0^=subkey[0];x1^=subkey[1]; ¥
                                x2^=subkey[2];x3^=subkey[3]

/* PHI: Constant used in the key schedule */
#define PHI 0x9e3779b9L

//////////////////////////////////////////////////////////////////////

// stdafx.h：標準のシステム インクルード ファイル、
//                 または参照回数が多く、かつあまり変更されない
//                 プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__DADF074B_67B7_49D8_AE87_84A20E3EE29B__INCLUDED_)
#define AFX_STDAFX_H__DADF074B_67B7_49D8_AE87_84A20E3EE29B__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif
// !defined(AFX_STDAFX_H__DADF074B_67B7_49D8_AE87_84A20E3EE29B__INCLUDED_)
```

１０．４．２　復号化部分

```
/* Copyright (C) 1998 Ross Anderson, Eli Biham, Lars Knudsen
 * All rights reserved.
 *
 * This code is freely distributed for AES selection process.
 * No other use is allowed.
 *
 * Copyright remains of the copyright holders, and as such any Copyright
 * notices in the code are not to be removed.
 *
```

```
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted only for the AES selection process, provided
 * that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *       notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed without the authors permission.
 *    i.e. this code cannot simply be copied and put under another distribution
 * licence [including the GNU Public Licence.]
 */

#include "stdafx.h"
#include "serpent.h"
#include "serpentsboxes.h"

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#define BLOCK_SIZE 128
#define DWORD unsigned long

/*    The functions    */
int serpent_makeKey(keyInstance *key, BYTE direction, int keyLen,
                                  char *keyMaterial)
{
   unsigned long i,j;
   unsigned long w[132],k[132];
   int rc;

   if(direction != DIR_ENCRYPT &&
        direction != DIR_DECRYPT)
      return BAD_KEY_DIR;
```

```
if(keyLen>256 || keyLen<1)
   return BAD_KEY_MAT;

key->direction=direction;
key->keyLen=keyLen;
strncpy(key->keyMaterial, keyMaterial, MAX_KEY_SIZE+1);

rc=serpent_convert_from_string(keyLen, keyMaterial,key->key);
if(rc<=0)
   return BAD_KEY_MAT;

for(i=0; i<keyLen/32; i++)
   w[i]=key->key[i];
if(keyLen<256)
   w[i]=(key->key[i]&((1L<<((keyLen&31)))-1))|(1L<<((keyLen&31)));
for(i++; i<8; i++)
   w[i]=0;
for(i=8; i<16; i++)
   w[i]=ROL(w[i-8]^w[i-5]^w[i-3]^w[i-1]^PHI^(i-8),11);
for(i=0; i<8; i++)
   w[i]=w[i+8];
for(i=8; i<132; i++)
   w[i]=ROL(w[i-8]^w[i-5]^w[i-3]^w[i-1]^PHI^i,11);

RND03(w[  0], w[  1], w[  2], w[  3], k[  0], k[  1], k[  2], k[  3]);
RND02(w[  4], w[  5], w[  6], w[  7], k[  4], k[  5], k[  6], k[  7]);
RND01(w[  8], w[  9], w[ 10], w[ 11], k[  8], k[  9], k[ 10], k[ 11]);
RND00(w[ 12], w[ 13], w[ 14], w[ 15], k[ 12], k[ 13], k[ 14], k[ 15]);
RND31(w[ 16], w[ 17], w[ 18], w[ 19], k[ 16], k[ 17], k[ 18], k[ 19]);
RND30(w[ 20], w[ 21], w[ 22], w[ 23], k[ 20], k[ 21], k[ 22], k[ 23]);
RND29(w[ 24], w[ 25], w[ 26], w[ 27], k[ 24], k[ 25], k[ 26], k[ 27]);
RND28(w[ 28], w[ 29], w[ 30], w[ 31], k[ 28], k[ 29], k[ 30], k[ 31]);
RND27(w[ 32], w[ 33], w[ 34], w[ 35], k[ 32], k[ 33], k[ 34], k[ 35]);
RND26(w[ 36], w[ 37], w[ 38], w[ 39], k[ 36], k[ 37], k[ 38], k[ 39]);
RND25(w[ 40], w[ 41], w[ 42], w[ 43], k[ 40], k[ 41], k[ 42], k[ 43]);
RND24(w[ 44], w[ 45], w[ 46], w[ 47], k[ 44], k[ 45], k[ 46], k[ 47]);
RND23(w[ 48], w[ 49], w[ 50], w[ 51], k[ 48], k[ 49], k[ 50], k[ 51]);
RND22(w[ 52], w[ 53], w[ 54], w[ 55], k[ 52], k[ 53], k[ 54], k[ 55]);
RND21(w[ 56], w[ 57], w[ 58], w[ 59], k[ 56], k[ 57], k[ 58], k[ 59]);
RND20(w[ 60], w[ 61], w[ 62], w[ 63], k[ 60], k[ 61], k[ 62], k[ 63]);
RND19(w[ 64], w[ 65], w[ 66], w[ 67], k[ 64], k[ 65], k[ 66], k[ 67]);
RND18(w[ 68], w[ 69], w[ 70], w[ 71], k[ 68], k[ 69], k[ 70], k[ 71]);
RND17(w[ 72], w[ 73], w[ 74], w[ 75], k[ 72], k[ 73], k[ 74], k[ 75]);
RND16(w[ 76], w[ 77], w[ 78], w[ 79], k[ 76], k[ 77], k[ 78], k[ 79]);
RND15(w[ 80], w[ 81], w[ 82], w[ 83], k[ 80], k[ 81], k[ 82], k[ 83]);
RND14(w[ 84], w[ 85], w[ 86], w[ 87], k[ 84], k[ 85], k[ 86], k[ 87]);
RND13(w[ 88], w[ 89], w[ 90], w[ 91], k[ 88], k[ 89], k[ 90], k[ 91]);
RND12(w[ 92], w[ 93], w[ 94], w[ 95], k[ 92], k[ 93], k[ 94], k[ 95]);
RND11(w[ 96], w[ 97], w[ 98], w[ 99], k[ 96], k[ 97], k[ 98], k[ 99]);
RND10(w[100], w[101], w[102], w[103], k[100], k[101], k[102], k[103]);
RND09(w[104], w[105], w[106], w[107], k[104], k[105], k[106], k[107]);
RND08(w[108], w[109], w[110], w[111], k[108], k[109], k[110], k[111]);
RND07(w[112], w[113], w[114], w[115], k[112], k[113], k[114], k[115]);
RND06(w[116], w[117], w[118], w[119], k[116], k[117], k[118], k[119]);
RND05(w[120], w[121], w[122], w[123], k[120], k[121], k[122], k[123]);
RND04(w[124], w[125], w[126], w[127], k[124], k[125], k[126], k[127]);
RND03(w[128], w[129], w[130], w[131], k[128], k[129], k[130], k[131]);

for(i=0; i<=32; i++)
```

```
      for(j=0; j<4; j++)
         key->subkeys[i][j] = k[4*i+j];

   return TRUE;
}

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV)
{
//   int i;
   int rc;

   if((mode != MODE_ECB) &&
      (mode != MODE_CBC) &&
      (mode != MODE_CFB1))
      return BAD_CIPHER_MODE;

   cipher->mode = mode;                /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
   cipher->blockSize=128;
   if(mode != MODE_ECB)
      {
         rc=serpent_convert_from_string(cipher->blockSize, IV, (unsigned long*)cipher->IV);
         if(rc<=0)
            return BAD_CIPHER_STATE;
      }

   return TRUE;
}

int blockEncrypt(cipherInstance *cipher,
                         keyInstance *key,
                         BYTE *input,
                         int inputLen,
                         BYTE *outBuffer)
{
   unsigned long t[4];
   int    b, n, i;
   DWORD x[BLOCK_SIZE/32];
   BYTE bit, bit0, ctBit, carry;

   /*
    * Note about optimization: the code becomes slower of the calls to
    * serpent_encrypt and serpent_decrypt are replaced by inlined code.
    * (tested on Pentium 133MMX)
    */

   switch(cipher->mode)
      {
      case MODE_ECB:
         for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
            serpent_encrypt((unsigned long*)input,(unsigned long*) outBuffer, key->subkeys);
         return inputLen;

      case MODE_CBC:
         t[0] = ((unsigned long*)cipher->IV)[0];
         t[1] = ((unsigned long*)cipher->IV)[1];
         t[2] = ((unsigned long*)cipher->IV)[2];
         t[3] = ((unsigned long*)cipher->IV)[3];
         for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
            {
```

```
            t[0] ^= ((unsigned long*)input)[0];
            t[1] ^= ((unsigned long*)input)[1];
            t[2] ^= ((unsigned long*)input)[2];
            t[3] ^= ((unsigned long*)input)[3];
            serpent_encrypt(t, t, key->subkeys);
            ((unsigned long*)outBuffer)[0] = t[0];
            ((unsigned long*)outBuffer)[1] = t[1];
            ((unsigned long*)outBuffer)[2] = t[2];
            ((unsigned long*)outBuffer)[3] = t[3];
          }
        ((unsigned long*)cipher->IV)[0] = t[0];
        ((unsigned long*)cipher->IV)[1] = t[1];
        ((unsigned long*)cipher->IV)[2] = t[2];
        ((unsigned long*)cipher->IV)[3] = t[3];

        return inputLen;

      case MODE_CFB1:
                    cipher->mode = MODE_ECB;        /* do encryption in ECB */
                    for (n=0;n<inputLen;n++)
                            {
                            blockEncrypt(cipher,key,(BYTE       *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                            bit0    = 0x80 >> (n & 7);/* which bit position in byte */
                            ctBit = (input[n/8] & bit0) ^ ((((BYTE *) x)[0] & 0x80) >> (n&7));
                            outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) | ctBit;
                            carry = ctBit >> (7 - (n&7));
                            for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                    {
                                    bit = cipher->IV[i] >> 7;     /* save next "carry" from shift */
                                    cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                    carry = bit;
                                    }
                            }
                    cipher->mode = MODE_CFB1;       /* restore mode for next time */
                    return inputLen;

      default:
        return BAD_CIPHER_STATE;
      }
}

int blockDecrypt(cipherInstance *cipher,
                        keyInstance *key,
                        BYTE *input,
                        int inputLen,
                        BYTE *outBuffer)
{
   unsigned long t[4];
   int    b, n, i;
   DWORD x[BLOCK_SIZE/32];
   BYTE bit, bit0, ctBit, carry;

   switch(cipher->mode)
     {
     case MODE_ECB:
       for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
            serpent_decrypt( (unsigned long*)input, (unsigned long*)outBuffer, key->subkeys);
       return inputLen;
```

```
    case MODE_CBC:
      t[0] = ((unsigned long*)cipher->IV)[0];
      t[1] = ((unsigned long*)cipher->IV)[1];
      t[2] = ((unsigned long*)cipher->IV)[2];
      t[3] = ((unsigned long*)cipher->IV)[3];
      for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
        {
          serpent_decrypt((unsigned long*)input, (unsigned long*)outBuffer,key->subkeys);
          ((unsigned long*)outBuffer)[0] ^= t[0];
          ((unsigned long*)outBuffer)[1] ^= t[1];
          ((unsigned long*)outBuffer)[2] ^= t[2];
          ((unsigned long*)outBuffer)[3] ^= t[3];
          t[0] = ((unsigned long*)input)[0];
          t[1] = ((unsigned long*)input)[1];
          t[2] = ((unsigned long*)input)[2];
          t[3] = ((unsigned long*)input)[3];
        }
      ((unsigned long*)cipher->IV)[0] = t[0];
      ((unsigned long*)cipher->IV)[1] = t[1];
      ((unsigned long*)cipher->IV)[2] = t[2];
      ((unsigned long*)cipher->IV)[3] = t[3];
      return inputLen;

    case MODE_CFB1://blockDecrypt
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,(BYTE       *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                        bit0    = 0x80 >> (n & 7);
                        ctBit = input[n/8] & bit0;
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) |
                                                                (ctBit ^ ((((BYTE *) x)[0] & 0x80)
>> (n&7)));
                        carry = ctBit >> (7 - (n&7));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;       /* restore mode for next time */
                return inputLen;

    default:
      return BAD_CIPHER_STATE;
    }
}

void serpent_encrypt(unsigned long plaintext[4],
                unsigned long ciphertext[4],
                unsigned long subkeys[33][4])
{
  register unsigned long x0, x1, x2, x3;
  register unsigned long y0, y1, y2, y3;

  x0=plaintext[0];
```

```
x1=plaintext[1];
x2=plaintext[2];
x3=plaintext[3];

/* Start to encrypt the plaintext x */
keying(x0, x1, x2, x3, subkeys[ 0]);
RND00(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 1]);
RND01(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 2]);
RND02(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 3]);
RND03(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 4]);
RND04(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 5]);
RND05(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 6]);
RND06(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 7]);
RND07(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 8]);
RND08(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 9]);
RND09(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[10]);
RND10(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[11]);
RND11(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[12]);
RND12(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[13]);
RND13(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[14]);
RND14(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[15]);
RND15(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[16]);
RND16(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[17]);
RND17(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
```

```
    keying(x0, x1, x2, x3, subkeys[18]);
    RND18(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[19]);
    RND19(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[20]);
    RND20(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[21]);
    RND21(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[22]);
    RND22(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[23]);
    RND23(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[24]);
    RND24(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[25]);
    RND25(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[26]);
    RND26(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[27]);
    RND27(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[28]);
    RND28(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[29]);
    RND29(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[30]);
    RND30(x0, x1, x2, x3, y0, y1, y2, y3);
    transform(y0, y1, y2, y3, x0, x1, x2, x3);
    keying(x0, x1, x2, x3, subkeys[31]);
    RND31(x0, x1, x2, x3, y0, y1, y2, y3);
    x0 = y0; x1 = y1; x2 = y2; x3 = y3;
    keying(x0, x1, x2, x3, subkeys[32]);
    /* The ciphertext is now in x */

    ciphertext[0] = x0;
    ciphertext[1] = x1;
    ciphertext[2] = x2;
    ciphertext[3] = x3;
}

void serpent_decrypt(unsigned long ciphertext[4],
                     unsigned long plaintext[4],
                     unsigned long subkeys[33][4])
{
    register unsigned long x0, x1, x2, x3;
    register unsigned long y0, y1, y2, y3;

    x0=ciphertext[0];
```

```
x1=ciphertext[1];
x2=ciphertext[2];
x3=ciphertext[3];

/* Start to decrypt the ciphertext x */
keying(x0, x1, x2, x3, subkeys[32]);
InvRND31(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[31]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND30(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[30]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND29(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[29]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND28(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[28]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND27(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[27]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND26(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[26]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND25(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[25]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND24(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[24]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND23(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[23]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND22(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[22]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND21(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[21]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND20(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[20]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND19(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[19]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND18(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[18]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND17(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[17]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND16(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[16]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND15(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[15]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND14(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[14]);
```

```
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND13(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[13]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND12(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[12]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND11(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[11]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND10(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[10]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND09(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 9]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND08(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 8]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND07(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 7]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND06(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 6]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND05(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 5]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND04(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 4]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND03(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 3]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND02(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 2]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND01(x0, x1, x2, x3, y0, y1, y2, y3);
    keying(y0, y1, y2, y3, subkeys[ 1]);
    inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
    InvRND00(x0, x1, x2, x3, y0, y1, y2, y3);
    x0 = y0; x1 = y1; x2 = y2; x3 = y3;
    keying(x0, x1, x2, x3, subkeys[ 0]);
    /* The plaintext is now in x */

    plaintext[0] = x0;
    plaintext[1] = x1;
    plaintext[2] = x2;
    plaintext[3] = x3;
}

#define min(x,y) (((x)<(y))?(x):(y))

int serpent_convert_from_string(int len, char *str, unsigned long *val)
/* the size of val must be at least the next multiple of 32 */
/* bits after len bits */
{
    int is, iv;
    int slen=min((int)strlen(str), (len+3)/4);
```

```
    if(len<0)
      return -1;                    /* Error!!! */

    if(len>slen*4 || len<slen*4-3)
      return -1;                    /* Error!!! */

    for(is=0; is<slen; is++)
      if(((str[is]<'0')||(str[is]>'9')) &&
         ((str[is]<'A')||(str[is]>'F')) &&
         ((str[is]<'a')||(str[is]>'f')))
        return -1;  /* Error!!! */

    for(is=slen, iv=0; is>=8; is-=8, iv++)
      {
        unsigned long t;
        sscanf(&str[is-8], "%08lX", &t);
        val[iv] = t;
      }
    if(is>0)
      {
        char tmp[10];
        unsigned long t;
        strncpy(tmp, str, is);
        tmp[is] = 0;
        sscanf(tmp, "%08lX", &t);
        val[iv++] = t;
      }
    for(; iv<(len+31)/32; iv++)
      val[iv] = 0;
    return iv;
}

char *serpent_convert_to_string(int len, unsigned long val[8], char *str)
/* str must have at least (len+3)/4+1 bytes. */
{
    int i;

    if(len<0)
      return (char *)-1;                    /* Error!!! */

    str[0] = 0;
    i=len/32;
    if(len&31>0)
      {
        char tmp[10];
        sprintf(tmp, "%08lX", val[i]&(((len&31)<<1)-1));
        strcat(str, &tmp[8-(((len&31)+3)/4)]);
      }
    for(i--; i>=0; i--)
      {
        char tmp[10];
        sprintf(tmp, "%08lX", val[i]);
        strcat(str, tmp);
      }
    return str;
}
```

```
//////////////////////////////////////////////////////

// SerpentDC.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include <stdio.h>
#include <time.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <malloc.h>
#include "serpent.h"

#define file_len(x) (unsigned long)x

int s;
keyInstance keyI;
cipherInstance cipherI;

int main(int argc, char* argv[])
{
	FILE *fkey , *fin , *fout ;
    fpos_t flen;
    unsigned long len, rlen, blen4, pfilelen;
	int mode,klen,blen,rc;
	char c_mode[4], c_klen[8], c_key[96], c_cini[64];
	unsigned char cbuf[2048], pbuf[2048];

	blen4 = 2048;

	printf( "Start!¥n" );

	if( argc != 4 ){                                    // 使い方の誤り
		printf( "argc != 4 ¥n" );
		return -1;
	}

	if((fkey = fopen(argv[1], "rt")) == NULL){
		printf("Can not find key file for encryption. ¥n");
		return (-1);
	}

	fgets( c_mode, 4, fkey );
	fgets( c_klen, 8, fkey );
	fgets( c_key, 96, fkey );
	fgets( c_cini, 64, fkey );

	mode = atoi(c_mode);
	klen = atoi(c_klen);
	blen = 128;

	if(klen<56 || 256<klen){
		printf("Wrong key size. ¥n");
		return (-1);
	}

	/*Set mode*/
```

```
        if(mode == 1){
                rc=cipherInit(&cipherI, MODE_ECB, "");
        }
        if(mode == 2){
                rc=cipherInit(&cipherI, MODE_CBC, c_cini);
        }
        if(mode == 3){
                rc=cipherInit(&cipherI, MODE_CFB1, c_cini);
        }
        if(rc<=0){
                printf("モード設定が出来ません。");
                return(-2);
        }
        serpent_makeKey(&keyI, DIR_DECRYPT,    klen, c_key );


        if((fin = fopen(argv[2], "rb")) == NULL){
                printf("Can not open crypted file. ¥n");
                return (-1);
        }
        if((fout = fopen(argv[3], "wb")) == NULL){
                printf("Can not open plane data file. ¥n");
                return (-1);
        }

    fseek(fin, 0, SEEK_END);
    fgetpos(fin, &flen);
    rlen = file_len(flen);
    // reset to start
    fseek(fin, 0, SEEK_SET);
        s = sizeof(unsigned long);
        // write the bytes of the file
        len = (unsigned long) fread(cbuf, 1, blen4 , fin);
        rlen = rlen - len;

        if(len < blen4){ return -3;  }

        rc=blockDecrypt(&cipherI,           &keyI,(unsigned           char*)(cbuf),8*blen4,(unsigned
char*)(pbuf));
                // 復号文出力
        pfilelen = *((long*)(pbuf));

        if(pfilelen <= blen4 - s){
                if(fwrite(pbuf+s, 1, pfilelen, fout) != pfilelen){return -2;}
                if(fkey){ fclose(fkey);         }
                if(fout){ fclose(fout);         }
                if(fin)     { fclose(fin);          }
                printf( "End!¥n" );
                return 0;
        }
        else{
                if(fwrite(pbuf+s, 1, blen4 - s, fout) != blen4 - s){return -2;}
                pfilelen -= (blen4 - s);
        }

    if((rlen <= blen4) && (rlen > 0))
        {           // if the file length is less than or equal to 2048 bytes
        len = (unsigned long) fread(cbuf, 1, blen4 , fin);
        rlen -= len;
```

```
        if(rlen > 0){ return -2; }
                rc=blockDecrypt(&cipherI,  &keyI,(unsigned  char*)(cbuf),8*blen4,(unsigned
char*)(pbuf));
                if(fwrite(pbuf, 1, pfilelen, fout) != pfilelen){ return -2; }
                if(fkey){ fclose(fkey);       }
                if(fout){ fclose(fout);       }
                if(fin)   { fclose(fin);      }
                printf( "End!¥n" );
                return 0;
    }
    else
    {   // if the file length is more 1024 bytes
        // read the file a block at a time
        while(rlen > 0 && !feof(fin))
        {
            // read a block and reduce the remaining byte count
            len = (unsigned long)fread(cbuf, 1, blen4 , fin);
            rlen -= len;
            if((rlen>0) && (len==blen4)){
                                rc=blockDecrypt(&cipherI,                    &keyI,(unsigned
char*)(cbuf),8*blen4,(unsigned char*)(pbuf));
                                if(fwrite(pbuf, 1, blen4, fout) != blen4){return -2;}
                                pfilelen -= blen4;
                        }
            if(rlen<=0){
                                rc=blockDecrypt(&cipherI,                    &keyI,(unsigned
char*)(cbuf),8*blen4,(unsigned char*)(pbuf));
                                if(fwrite(pbuf, 1, pfilelen, fout) != pfilelen){return -2;}
                                if(fkey){ fclose(fkey);       }
                                if(fout){ fclose(fout);       }
                                if(fin)   { fclose(fin);      }
                                printf( "End!¥n" );
                                return 0;
                        }
        }
    }
        if(fkey){ fclose(fkey);       }
        if(fout){ fclose(fout);       }
        if(fin)   { fclose(fin);      }
        printf( "End!¥n" );
        return 0;
}

//////////////////////////////////////////////////////

// stdafx.cpp：標準インクルードファイルを含むソース ファイル
//                  SerpentDC.pch 生成されるプリコンパイル済ヘッダー
//                  stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。


//////////////////////////////////////////////////////

/*  aes.h  */
```

```
/*  AES Cipher header file for ANSI C Submissions
      Lawrence E. Bassham III
      Computer Security Division
      National Institute of Standards and Technology

      April 15, 1998

    This sample is to assist implementers developing to the Cryptographic
API Profile for AES Candidate Algorithm Submissions.  Please consult this
document as a cross-reference.

    ANY CHANGES, WHERE APPROPRIATE, TO INFORMATION PROVIDED IN THIS FILE
MUST BE DOCUMENTED.  CHANGES ARE ONLY APPROPRIATE WHERE SPECIFIED WITH
THE STRING "CHANGE POSSIBLE".  FUNCTION CALLS AND THEIR PARAMETERS CANNOT
BE CHANGED.  STRUCTURES CAN BE ALTERED TO ALLOW IMPLEMENTERS TO INCLUDE
IMPLEMENTATION SPECIFIC INFORMATION.
*/

/*  Includes:
      Standard include files
*/

#include <stdio.h>

/*  Defines:
      Add any additional defines you need
*/

#define     DIR_ENCRYPT     0    /*  Are we encrpyting?  */
#define     DIR_DECRYPT     1    /*  Are we decrpyting?  */
#define     MODE_ECB        1    /*  Are we ciphering in ECB mode?    */
#define     MODE_CBC        2    /*  Are we ciphering in CBC mode?    */
#define     MODE_CFB1       3    /*  Are we ciphering in 1-bit CFB mode? */
#define     TRUE            1
#define     FALSE           0

/*  Error Codes - CHANGE POSSIBLE: inclusion of additional error codes  */
#define     BAD_KEY_DIR         -1  /*  Key direction is invalid, e;g;,
                                        unknown value */
#define     BAD_KEY_MAT         -2  /*  Key material not of correct
                                        length */
#define     BAD_KEY_INSTANCE    -3  /*  Key passed is not valid  */
#define     BAD_CIPHER_MODE     -4  /*  Params struct passed to
                                        cipherInit invalid */
#define     BAD_CIPHER_STATE    -5  /*  Cipher in wrong state (e.g., not
                                        initialized) */

/*  CHANGE POSSIBLE:  inclusion of algorithm specific defines  */
#define     MAX_KEY_SIZE    64  /* # of ASCII char's needed to
                                   represent a key */
#define     MAX_IV_SIZE         32  /* # of ASCII char's needed to
                                   represent an IV  */

/*  Typedefs:

      Typedef'ed data storage elements.  Add any algorithm specific
```

```
parameters at the bottom of the structs as appropriate.
*/

typedef      unsigned char      BYTE;

/*   The structure for key information */
typedef struct {
        BYTE   direction;   /*   Key used for encrypting or decrypting? */
        int    keyLen;      /*   Length of the key   */
        char   keyMaterial[MAX_KEY_SIZE+1];   /*   Raw key data in ASCII, e.g.,
                                                what the user types or KAT values)*/
        /*   The following parameters are algorithm dependent, replace or
                add as necessary   */
        unsigned long key[8];                   /* The key in binary */
        unsigned long subkeys[33][4];           /* Serpent subkeys */
        } keyInstance;

/*   The structure for cipher information */
typedef struct {
        BYTE       mode;               /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
        char   IV[MAX_IV_SIZE];    /* A possible Initialization Vector for
                                                ciphering */
        /*   Add any algorithm specific parameters needed here   */
        int    blockSize;              /* Sample: Handles non-128 bit block sizes
                                                (if available) */
        } cipherInstance;


/*   Function protoypes   */
int serpent_makeKey(keyInstance *key, BYTE direction, int keyLen,
                char *keyMaterial);

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                        int inputLen, BYTE *outBuffer);

int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                        int inputLen, BYTE *outBuffer);

int serpent_convert_from_string(int, char *, unsigned long *);
void serpent_encrypt(unsigned long [],unsigned long [], unsigned long [][4]);
void serpent_decrypt(unsigned long [],unsigned long [], unsigned long [][4]);


//////////////////////////////////////////////////////

/* Copyright (C) 1998 Ross Anderson, Eli Biham, Lars Knudsen
 * All rights reserved.
 *
 * This code is freely distributed for AES selection process.
 * No other use is allowed.
 *
 * Copyright remains of the copyright holders, and as such any Copyright
 * notices in the code are not to be removed.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted only for the AES selection process, provided
 * that the following conditions
```

```
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *       notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed without the authors permission.
 *   i.e. this code cannot simply be copied and put under another distribution
 * licence [including the GNU Public Licence.]
 */

/* S0:   3   8 15   1 10   6   5 11 14 13   4   2   7   0   9 12 */

/* depth = 5,7,4,2, Total gates=18 */
#define RND00(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t05, t06, t07, t08, t09, t11, t12, t13, t14, t15, t17, t01;¥
        t01 = b     ^ c   ; ¥
        t02 = a     | d   ; ¥
        t03 = a     ^ b   ; ¥
        z     = t02 ^ t01; ¥
        t05 = c     | z   ; ¥
        t06 = a     ^ d   ; ¥
        t07 = b     | c   ; ¥
        t08 = d     & t05; ¥
        t09 = t03 & t07; ¥
        y     = t09 ^ t08; ¥
        t11 = t09 & y   ; ¥
        t12 = c     ^ d   ; ¥
        t13 = t07 ^ t11; ¥
        t14 = b     & t06; ¥
        t15 = t06 ^ t13; ¥
        w     =       ~ t15; ¥
        t17 = w     ^ t14; ¥
        x     = t12 ^ t17; }

/* InvS0:   13   3 11   0 10   6   5 12   1 14   4   7 15   9   8   2 */

/* depth = 8,4,3,6, Total gates=19 */
#define InvRND00(a,b,c,d,w,x,y,z) ¥
```

```
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t12, t13, t14, t15, t17, t18,
t01;¥
        t01 = c     ^ d   ; ¥
        t02 = a     | b   ; ¥
        t03 = b     | c   ; ¥
        t04 = c     & t01; ¥
        t05 = t02 ^ t01; ¥
        t06 = a     | t04; ¥
        y     =       ~ t05; ¥
        t08 = b     ^ d   ; ¥
        t09 = t03 & t08; ¥
        t10 = d     | y   ; ¥
        x     = t09 ^ t06; ¥
        t12 = a     | t05; ¥
        t13 = x     ^ t12; ¥
        t14 = t03 ^ t10; ¥
        t15 = a     ^ c   ; ¥
        z     = t14 ^ t13; ¥
        t17 = t05 & t13; ¥
        t18 = t14 | t17; ¥
        w     = t15 ^ t18; }

/* S1:   15 12   2   7   9   0   5 10   1 11 14   8   6 13   3   4 */

/* depth = 10,7,3,5, Total gates=18 */
#define RND01(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t10, t11, t12, t13, t16, t17, t01;¥
        t01 = a     | d   ; ¥
        t02 = c     ^ d   ; ¥
        t03 =       ~ b   ; ¥
        t04 = a     ^ c   ; ¥
        t05 = a     | t03; ¥
        t06 = d     & t04; ¥
        t07 = t01 & t02; ¥
        t08 = b     | t06; ¥
        y     = t02 ^ t05; ¥
        t10 = t07 ^ t08; ¥
        t11 = t01 ^ t10; ¥
        t12 = y     ^ t11; ¥
        t13 = b     & d   ; ¥
        z     =       ~ t10; ¥
        x     = t13 ^ t12; ¥
        t16 = t10 | x   ; ¥
        t17 = t05 & t16; ¥
        w     = c     ^ t17; }

/* InvS1:    5   8   2 14 15   6 12   3 11   4   7   9   1 13 10   0 */

/* depth = 7,4,5,3, Total gates=18 */
#define InvRND01(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t14, t15, t17, t01;¥
        t01 = a     ^ b   ; ¥
        t02 = b     | d   ; ¥
        t03 = a     & c   ; ¥
        t04 = c     ^ t02; ¥
        t05 = a     | t04; ¥
        t06 = t01 & t05; ¥
        t07 = d     | t03; ¥
        t08 = b     ^ t06; ¥
```

```
        t09 = t07 ^ t06; ¥
        t10 = t04 | t03; ¥
        t11 = d   & t08; ¥
        y   =     ~ t09; ¥
        x   = t10 ^ t11; ¥
        t14 = a   | y  ; ¥
        t15 = t06 ^ x  ; ¥
        z   = t01 ^ t04; ¥
        t17 = c   ^ t15; ¥
        w   = t14 ^ t17; }

/* S2:   8  6  7  9  3 12 10 15 13  1 14  4  0 11  5  2 */

/* depth = 3,8,11,7, Total gates=16 */
#define RND02(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t05, t06, t07, t08, t09, t10, t12, t13, t14, t01;¥
        t01 = a   | c  ; ¥
        t02 = a   ^ b  ; ¥
        t03 = d   ^ t01; ¥
        w   = t02 ^ t03; ¥
        t05 = c   ^ w  ; ¥
        t06 = b   ^ t05; ¥
        t07 = b   | t05; ¥
        t08 = t01 & t06; ¥
        t09 = t03 ^ t07; ¥
        t10 = t02 | t09; ¥
        x   = t10 ^ t08; ¥
        t12 = a   | d  ; ¥
        t13 = t09 ^ x  ; ¥
        t14 = b   ^ t13; ¥
        z   =     ~ t09; ¥
        y   = t12 ^ t14; }

/* InvS2:  12  9 15  4 11 14  1  2  0  3  6 13  5  8 10  7 */

/* depth = 3,6,8,3, Total gates=18 */
#define InvRND02(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t06, t07, t08, t09, t10, t11, t12, t15, t16, t17, t01;¥
        t01 = a   ^ d  ; ¥
        t02 = c   ^ d  ; ¥
        t03 = a   & c  ; ¥
        t04 = b   | t02; ¥
        w   = t01 ^ t04; ¥
        t06 = a   | c  ; ¥
        t07 = d   | w  ; ¥
        t08 =     ~ d  ; ¥
        t09 = b   & t06; ¥
        t10 = t08 | t03; ¥
        t11 = b   & t07; ¥
        t12 = t06 & t02; ¥
        z   = t09 ^ t10; ¥
        x   = t12 ^ t11; ¥
        t15 = c   & z  ; ¥
        t16 = w   ^ x  ; ¥
        t17 = t10 ^ t15; ¥
        y   = t16 ^ t17; }

/* S3:   0 15 11  8 12  9  6  3 13  1  2  4 10  7  5 14 */
```

```
/* depth = 8,3,5,5, Total gates=18 */
#define RND03(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t13, t14, t15, t01;¥
        t01 = a     ^ c   ; ¥
        t02 = a     | d   ; ¥
        t03 = a     & d   ; ¥
        t04 = t01 & t02; ¥
        t05 = b     | t03; ¥
        t06 = a     & b   ; ¥
        t07 = d     ^ t04; ¥
        t08 = c     | t06; ¥
        t09 = b     ^ t07; ¥
        t10 = d     & t05; ¥
        t11 = t02 ^ t10; ¥
        z     = t08 ^ t09; ¥
        t13 = d     | z   ; ¥
        t14 = a     | t07; ¥
        t15 = b     & t13; ¥
        y     = t08 ^ t11; ¥
        w     = t14 ^ t15; ¥
        x     = t05 ^ t04; }

/* InvS3:     0    9 10    7 11 14    6 13    3    5 12    2    4    8 15    1 */

/* depth = 3,6,4,4, Total gates=17 */
#define InvRND03(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t09, t11, t12, t13, t14, t16, t01;¥
        t01 = c     | d   ; ¥
        t02 = a     | d   ; ¥
        t03 = c     ^ t02; ¥
        t04 = b     ^ t02; ¥
        t05 = a     ^ d   ; ¥
        t06 = t04 & t03; ¥
        t07 = b     & t01; ¥
        y     = t05 ^ t06; ¥
        t09 = a     ^ t03; ¥
        w     = t07 ^ t03; ¥
        t11 = w     | t05; ¥
        t12 = t09 & t11; ¥
        t13 = a     & y   ; ¥
        t14 = t01 ^ t05; ¥
        x     = b     ^ t12; ¥
        t16 = b     | t13; ¥
        z     = t14 ^ t16; }

/* S4:     1 15    8    3 12    0 11    6    2    5    4 10    9 14    7 13 */

/* depth = 6,7,5,3, Total gates=19 */
#define RND04(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t11, t12, t13, t14, t15, t16,
t01;¥
        t01 = a     | b   ; ¥
        t02 = b     | c   ; ¥
        t03 = a     ^ t02; ¥
        t04 = b     ^ d   ; ¥
        t05 = d     | t03; ¥
        t06 = d     & t01; ¥
        z     = t03 ^ t06; ¥
        t08 = z     & t04; ¥
```

```
        t09 = t04 & t05; ¥
        t10 = c     ^ t06; ¥
        t11 = b     & c   ; ¥
        t12 = t04 ^ t08; ¥
        t13 = t11 | t03; ¥
        t14 = t10 ^ t09; ¥
        t15 = a     & t05; ¥
        t16 = t11 | t12; ¥
        y     = t13 ^ t08; ¥
        x     = t15 ^ t16; ¥
        w     =       ~ t14; }

/* InvS4:    5   0   8   3 10   9   7 14   2 12 11   6   4 15 13   1 */

/* depth = 6,4,7,3, Total gates=17 */
#define InvRND04(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t09, t10, t11, t12, t13, t15, t01;¥
        t01 = b     | d   ; ¥
        t02 = c     | d   ; ¥
        t03 = a     & t01; ¥
        t04 = b     ^ t02; ¥
        t05 = c     ^ d   ; ¥
        t06 =       ~ t03; ¥
        t07 = a     & t04; ¥
        x     = t05 ^ t07; ¥
        t09 = x     | t06; ¥
        t10 = a     ^ t07; ¥
        t11 = t01 ^ t09; ¥
        t12 = d     ^ t04; ¥
        t13 = c     | t10; ¥
        z     = t03 ^ t12; ¥
        t15 = a     ^ t04; ¥
        y     = t11 ^ t13; ¥
        w     = t15 ^ t09; }

/* S5:   15   5   2 11   4 10   9 12   0   3 14   8 13   6   7   1 */

/* depth = 4,6,8,6, Total gates=17 */
#define RND05(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t14, t01;¥
        t01 = b     ^ d   ; ¥
        t02 = b     | d   ; ¥
        t03 = a     & t01; ¥
        t04 = c     ^ t02; ¥
        t05 = t03 ^ t04; ¥
        w     =       ~ t05; ¥
        t07 = a     ^ t01; ¥
        t08 = d     | w   ; ¥
        t09 = b     | t05; ¥
        t10 = d     ^ t08; ¥
        t11 = b     | t07; ¥
        t12 = t03 | w   ; ¥
        t13 = t07 | t10; ¥
        t14 = t01 ^ t11; ¥
        y     = t09 ^ t13; ¥
        x     = t07 ^ t08; ¥
        z     = t12 ^ t14; }

/* InvS5:    8 15   2   9   4   1 13 14 11   6   5   3   7 12 10   0 */
```

```
/* depth = 4,6,9,7, Total gates=17 */
#define InvRND05(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t12, t13, t15, t16, t01;¥
        t01 = a     & d   ; ¥
        t02 = c     ^ t01; ¥
        t03 = a     ^ d   ; ¥
        t04 = b     & t02; ¥
        t05 = a     & c   ; ¥
        w     = t03 ^ t04; ¥
        t07 = a     & w   ; ¥
        t08 = t01 ^ w   ; ¥
        t09 = b     | t05; ¥
        t10 =       ~ b   ; ¥
        x     = t08 ^ t09; ¥
        t12 = t10 | t07; ¥
        t13 = w     | x   ; ¥
        z     = t02 ^ t12; ¥
        t15 = t02 ^ t13; ¥
        t16 = b     ^ d   ; ¥
        y     = t16 ^ t15; }

/* S6:    7   2 12   5   8   4   6 11 14   9   1 15 13   3 10   0 */

/* depth = 8,3,6,3, Total gates=19 */
#define RND06(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t15, t17, t18,
t01;¥
        t01 = a     & d   ; ¥
        t02 = b     ^ c   ; ¥
        t03 = a     ^ d   ; ¥
        t04 = t01 ^ t02; ¥
        t05 = b     | c   ; ¥
        x     =       ~ t04; ¥
        t07 = t03 & t05; ¥
        t08 = b     & x   ; ¥
        t09 = a     | c   ; ¥
        t10 = t07 ^ t08; ¥
        t11 = b     | d   ; ¥
        t12 = c     ^ t11; ¥
        t13 = t09 ^ t10; ¥
        y     =       ~ t13; ¥
        t15 = x     & t03; ¥
        z     = t12 ^ t07; ¥
        t17 = a     ^ b   ; ¥
        t18 = y     ^ t15; ¥
        w     = t17 ^ t18; }

/* InvS6:   15 10   1 13   5   3   6   0   4   9 14   7   2 12   8 11 */

/* depth = 5,3,8,6, Total gates=19 */
#define InvRND06(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t12, t13, t14, t15, t16, t17,
t01;¥
        t01 = a     ^ c   ; ¥
        t02 =       ~ c   ; ¥
        t03 = b     & t01; ¥
        t04 = b     | t02; ¥
        t05 = d     | t03; ¥
```

```
        t06 = b     ^ d   ; ¥
        t07 = a     & t04; ¥
        t08 = a     | t02; ¥
        t09 = t07 ^ t05; ¥
        x     = t06 ^ t08; ¥
        w     =       ~ t09; ¥
        t12 = b     & w   ; ¥
        t13 = t01 & t05; ¥
        t14 = t01 ^ t12; ¥
        t15 = t07 ^ t13; ¥
        t16 = d     | t02; ¥
        t17 = a     ^ x   ; ¥
        z     = t17 ^ t15; ¥
        y     = t16 ^ t14; }

/* S7:    1 13 15   0 14   8   2 11   7   4 12 10   9   3   5   6 */

/* depth = 10,7,10,4, Total gates=19 */
#define RND07(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t11, t13, t14, t15, t16, t17,
t01;¥
        t01 = a     & c   ; ¥
        t02 =       ~ d   ; ¥
        t03 = a     & t02; ¥
        t04 = b     | t01; ¥
        t05 = a     & b   ; ¥
        t06 = c     ^ t04; ¥
        z     = t03 ^ t06; ¥
        t08 = c     | z   ; ¥
        t09 = d     | t05; ¥
        t10 = a     ^ t08; ¥
        t11 = t04 & z   ; ¥
        x     = t09 ^ t10; ¥
        t13 = b     ^ x   ; ¥
        t14 = t01 ^ x   ; ¥
        t15 = c     ^ t05; ¥
        t16 = t11 | t13; ¥
        t17 = t02 | t14; ¥
        w     = t15 ^ t17; ¥
        y     = a     ^ t16; }

/* InvS7:    3   0   6 13   9 14 15   8   5 12 11   7 10   1   4   2 */

/* depth = 9,7,3,3, Total gates=18 */
#define InvRND07(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t06, t07, t08, t09, t10, t11, t13, t14, t15, t16, t01;¥
        t01 = a     & b   ; ¥
        t02 = a     | b   ; ¥
        t03 = c     | t01; ¥
        t04 = d     & t02; ¥
        z     = t03 ^ t04; ¥
        t06 = b     ^ t04; ¥
        t07 = d     ^ z   ; ¥
        t08 =       ~ t07; ¥
        t09 = t06 | t08; ¥
        t10 = b     ^ d   ; ¥
        t11 = a     | d   ; ¥
        x     = a     ^ t09; ¥
        t13 = c     ^ t06; ¥
```

```
        t14 = c     & t11; ¥
        t15 = d     | x   ; ¥
        t16 = t01 | t10; ¥
        w     = t13 ^ t15; ¥
        y     = t14 ^ t16; }

#define RND08(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND09(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND10(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND11(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND12(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND13(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND14(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND15(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)
#define RND16(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND17(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND18(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND19(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND20(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND21(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND22(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND23(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)
#define RND24(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND25(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND26(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND27(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND28(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND29(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND30(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND31(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)

#define InvRND08(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND09(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND10(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND11(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND12(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND13(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND14(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND15(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)
#define InvRND16(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND17(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND18(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND19(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND20(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND21(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND22(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND23(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)
#define InvRND24(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND25(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND26(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND27(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND28(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND29(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND30(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND31(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)

/* Linear transformations and key mixing: */

#define ROL(x,n) ((((unsigned long)(x))<<(n)) | ¥
```

```
                        (((unsigned long)(x))>>(32-(n))))
#define ROR(x,n) ((((unsigned long)(x))<<(32-(n))) | ¥
                        (((unsigned long)(x))>>(n)))

#define transform(x0, x1, x2, x3, y0, y1, y2, y3) ¥
        y0 = ROL(x0, 13); ¥
        y2 = ROL(x2, 3); ¥
        y1 = x1 ^ y0 ^ y2; ¥
        y3 = x3 ^ y2 ^ ((unsigned long)y0)<<3; ¥
        y1 = ROL(y1, 1); ¥
        y3 = ROL(y3, 7); ¥
        y0 = y0 ^ y1 ^ y3; ¥
        y2 = y2 ^ y3 ^ ((unsigned long)y1<<7); ¥
        y0 = ROL(y0, 5); ¥
        y2 = ROL(y2, 22)

#define inv_transform(x0, x1, x2, x3, y0, y1, y2, y3) ¥
        y2 = ROR(x2, 22);¥
        y0 = ROR(x0, 5); ¥
        y2 = y2 ^ x3 ^ ((unsigned long)x1<<7); ¥
        y0 = y0 ^ x1 ^ x3; ¥
        y3 = ROR(x3, 7); ¥
        y1 = ROR(x1, 1); ¥
        y3 = y3 ^ y2 ^ ((unsigned long)y0)<<3; ¥
        y1 = y1 ^ y0 ^ y2; ¥
        y2 = ROR(y2, 3); ¥
        y0 = ROR(y0, 13)

#define keying(x0, x1, x2, x3, subkey) ¥
                                x0^=subkey[0];x1^=subkey[1]; ¥
                                x2^=subkey[2];x3^=subkey[3]

/* PHI: Constant used in the key schedule */
#define PHI 0x9e3779b9L


////////////////////////////////////////////////////////////

// stdafx.h : 標準のシステム インクルード ファイル、
//                 または参照回数が多く、かつあまり変更されない
//                 プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__08F0068B_7015_4F63_A50A_35A9784E5640__INCLUDED_)
#define AFX_STDAFX_H__08F0068B_7015_4F63_A50A_35A9784E5640__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。
```

```
#endif
// !defined(AFX_STDAFX_H__08F0068B_7015_4F63_A50A_35A9784E5640__INCLUDED_)
```

１０．４．３　鍵の形式

鍵は次のような 3 行からなるテキストファイルです。

```
128
128
40B6E45ED3FE4CA2D1E14EDD9C48FC68
```

1 行目は、鍵の長さ
2 行目は、ブロックサイズ
3 行目は鍵を 16 進数で表したものです。

１０．５　MARS 暗号ソースコード

MARS 暗号は IBM が次の規定のもとにソースコードを公開している暗号です。これに従ってソースコードを公開しますが、この IBM の規定が適用されるのは、この MARS 暗号の部分だけです。

IBM Common Public License - V1.0

03 Mar 2005

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

1. in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
2. in the case of each subsequent Contributor:
    1. changes to the Program, and
    2. additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

1. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

2. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

3. Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

4. Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

1. it complies with the terms and conditions of this Agreement; and
2. its license agreement:
1. effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

2. effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
3. states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and
4. states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

1. it must be made available under this Agreement; and
2. a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial

Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

１０．５．１　暗号化部分

```
/* mars.c optimized C code - copyright(c) 1998 IBM
 *
 * This code implements both the NIST high level C API (version 5)
 * and an underlying IBM defined low level (DWORD oriented) C API.
 */

/* Revisions log:
 *
 *     Aug 1999, Shai - the "tweaked" key schedule
 *     Apr 1998, Dave && Shai - new key scheduling, new sbox, NIST API
 *     Mar 1998, Shai Halevi    - adapted to the latest variant of mars
 *     Feb 1998, Dave Safford - created
 */

/* Compilation using pcg's version of gcc:
 *       gcc -Wall -pedantic -c -O6 -fomit-frame-pointer -mcpu=pentiumpro
 *             -DINTEL_GCC tests.c mars-opt.c
 *
 * Compilation using xlc on AIX:
 *       xlc -c -O3 -DAIX_XLC mars-opt.c
 *
 * Compilation using Borland C++ 5.0 from a DOS command line:
 *       bcc32 -Oi -6 -v -A -a4 -O2 -DKAT tests.c mars-opt.c
 *
 * Useful compilation defines:
 *       NO_MIX         - do not execute the mixing phases (core only)
 *       INTEL_GCC    - optimized for pentiumpro and djgpp/gcc compiler
 *       AIX_XLC       - optimized for powerpc/AIX with xlc compiler
 *       SWAP_BYTES - force endian conversion (eg __BYTE_ORDER not supported)
 *       IVT              - add intermediate values test outputs (this slows
 *                          down encryption and decryption significantly)
```

```
 */
  #include "stdafx.h"
#include "aes.h"

/* The low level mars routines are completely DWORD oriented, and
 * endian neutral. The high level NIST routines provide BYTE oriented
 * inputs and outputs, thus raising the endian issue when converting
 * between BYTEs and DWORDs. For these conversions, mars assumes
 * little endian order. On a big endian machine, we define BSWAP()
 * to do the conversions. BSWAP() is about the best you can do in C.
 * Real implementations will undoubtedly use inline ASM, as most risc
 * machines can do this in one instruction.
 *
 * Make a best guess on platform endianness; This works on linux, AIX,
 * and W95. For other platforms, you may have to manually define SWAP_BYTES
 * on a big endian machine if this guessing doesn't work.
 */

#ifdef _AIX
#define SWAP_BYTES
#else
#ifdef __linux__
#include <endian.h>
#ifdef __BYTE_ORDER
#if __BYTE_ORDER == __BIG_ENDIAN
#define SWAP_BYTES
#endif
#endif
#endif
#endif
#ifdef SWAP_BYTES
#define BSWAP(x) ¥
      ( (( x                ) << 24) | ¥
        (((x)&0xff00    ) << 8 ) | ¥
        (((x)&0xff0000) >> 8 ) | ¥
        (( x                ) >> 24)    )
#else
#define BSWAP(x) (x)
#endif

/* Some compiler optimizers will recognize certain rotation idioms,
 * and reduce them to native rotation instructions. These idioms
 * have been found to work for GCC on Intel, and XLC on RS6000.
 * Nothing seems to work on Borland C++ 5.0, although we can leave
 * out the mask on 'b'.
 */
#ifdef INTEL_GCC
#define LROTATE(a,b) (((a)<<(int)(b)) | ((a)>>(W - (int)(b))))
#define RROTATE(a,b) (((a)>>(int)(b)) | ((a)<<(W - (int)(b))))
#else
#ifdef AIX_XLC
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#ifdef __BORLANDC__
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#define LROTATE(a,b) (((a)<<(int)(b&31)) | ((a)>>(W - (int)(b&31))))
```

```
#define RROTATE(a,b) (((a)>>(int)(b&31)) | ((a)<<(W - (int)(b&31))))
#endif
#endif
#endif

//////////////////////////////////////////
#define BLOCK_SIZE 128
   unsigned long t[4];
   int    b, n, i;
   DWORD x[BLOCK_SIZE/32];
   BYTE bit, bit0, ctBit, carry;
/* two 8 x 32 sboxes - stored together to save a pointer
 * these sboxes were generated with buf[3] = 0x02917d59, as
 * chosen by sbox.c, which had the following output:
 *     After 54817140, (38023 fail) new min j = 43089241 (0x2917d59)
 *     test 0 eval 0.044922 (single bit correlation)
 *     test 1 eval 0.033203 (single bit bias)
 *     test 2 eval 0.031250 (consecutive bit bias)
 *     test 3 eval 0.007813 (parity bias)
 *     test 4 eval 0.148438 (avalanche)
 */
static DWORD S[512] = {
   0x09d0c479, 0x28c8ffe0, 0x84aa6c39, 0x9dad7287,
   0x7dff9be3, 0xd4268361, 0xc96da1d4, 0x7974cc93,
   0x85d0582e, 0x2a4b5705, 0x1ca16a62, 0xc3bd279d,
   0x0f1f25e5, 0x5160372f, 0xc695c1fb, 0x4d7ff1e4,
   0xae5f6bf4, 0x0d72ee46, 0xff23de8a, 0xb1cf8e83,
   0xf14902e2, 0x3e981e42, 0x8bf53eb6, 0x7f4bf8ac,
   0x83631f83, 0x25970205, 0x76afe784, 0x3a7931d4,
   0x4f846450, 0x5c64c3f6, 0x210a5f18, 0xc6986a26,
   0x28f4e826, 0x3a60a81c, 0xd340a664, 0x7ea820c4,
   0x526687c5, 0x7eddd12b, 0x32a11d1d, 0x9c9ef086,
   0x80f6e831, 0xab6f04ad, 0x56fb9b53, 0x8b2e095c,
   0xb68556ae, 0xd2250b0d, 0x294a7721, 0xe21fb253,
   0xae136749, 0xe82aae86, 0x93365104, 0x99404a66,
   0x78a784dc, 0xb69ba84b, 0x04046793, 0x23db5c1e,
   0x46cae1d6, 0x2fe28134, 0x5a223942, 0x1863cd5b,
   0xc190c6e3, 0x07dfb846, 0x6eb88816, 0x2d0dcc4a,
   0xa4ccae59, 0x3798670d, 0xcbfa9493, 0x4f481d45,
   0xeafc8ca8, 0xdb1129d6, 0xb0449e20, 0x0f5407fb,
   0x6167d9a8, 0xd1f45763, 0x4daa96c3, 0x3bec5958,
   0xababa014, 0xb6ccd201, 0x38d6279f, 0x02682215,
   0x8f376cd5, 0x092c237e, 0xbfc56593, 0x32889d2c,
   0x854b3e95, 0x05bb9b43, 0x7dcd5dcd, 0xa02e926c,
   0xfae527e5, 0x36a1c330, 0x3412e1ae, 0xf257f462,
   0x3c4f1d71, 0x30a2e809, 0x68e5f551, 0x9c61ba44,
   0x5ded0ab8, 0x75ce09c8, 0x9654f93e, 0x698c0cca,
   0x243cb3e4, 0x2b062b97, 0x0f3b8d9e, 0x00e050df,
   0xfc5d6166, 0xe35f9288, 0xc079550d, 0x0591aee8,
   0x8e531e74, 0x75fe3578, 0x2f6d829a, 0xf60b21ae,
   0x95e8eb8d, 0x6699486b, 0x901d7d9b, 0xfd6d6e31,
   0x1090acef, 0xe0670dd8, 0xdab2e692, 0xcd6d4365,
   0xe5393514, 0x3af345f0, 0x6241fc4d, 0x460da3a3,
   0x7bcf3729, 0x8bf1d1e0, 0x14aac070, 0x1587ed55,
   0x3afd7d3e, 0xd2f29e01, 0x29a9d1f6, 0xefb10c53,
   0xcf3b870f, 0xb414935c, 0x664465ed, 0x024acac7,
   0x59a744c1, 0x1d2936a7, 0xdc580aa6, 0xcf574ca8,
   0x040a7a10, 0x6cd81807, 0x8a98be4c, 0xaccea063,
   0xc33e92b5, 0xd1e0e03d, 0xb322517e, 0x2092bd13,
```

0x386b2c4a, 0x52e8dd58, 0x58656dfb, 0x50820371,
0x41811896, 0xe337ef7e, 0xd39fb119, 0xc97f0df6,
0x68fea01b, 0xa150a6e5, 0x55258962, 0xeb6ff41b,
0xd7c9cd7a, 0xa619cd9e, 0xbcf09576, 0x2672c073,
0xf003fb3c, 0x4ab7a50b, 0x1484126a, 0x487ba9b1,
0xa64fc9c6, 0xf6957d49, 0x38b06a75, 0xdd805fcd,
0x63d094cf, 0xf51c999e, 0x1aa4d343, 0xb8495294,
0xce9f8e99, 0xbffcd770, 0xc7c275cc, 0x378453a7,
0x7b21be33, 0x397f41bd, 0x4e94d131, 0x92cc1f98,
0x5915ea51, 0x99f861b7, 0xc9980a88, 0x1d74fd5f,
0xb0a495f8, 0x614deed0, 0xb5778eea, 0x5941792d,
0xfa90c1f8, 0x33f824b4, 0xc4965372, 0x3ff6d550,
0x4ca5fec0, 0x8630e964, 0x5b3fbbd6, 0x7da26a48,
0xb203231a, 0x04297514, 0x2d639306, 0x2eb13149,
0x16a45272, 0x532459a0, 0x8e5f4872, 0xf966c7d9,
0x07128dc0, 0x0d44db62, 0xafc8d52d, 0x06316131,
0xd838e7ce, 0x1bc41d00, 0x3a2e8c0f, 0xea83837e,
0xb984737d, 0x13ba4891, 0xc4f8b949, 0xa6d6acb3,
0xa215cdce, 0x8359838b, 0x6bd1aa31, 0xf579dd52,
0x21b93f93, 0xf5176781, 0x187dfdde, 0xe94aeb76,
0x2b38fd54, 0x431de1da, 0xab394825, 0x9ad3048f,
0xdfea32aa, 0x659473e3, 0x623f7863, 0xf3346c59,
0xab3ab685, 0x3346a90b, 0x6b56443e, 0xc6de01f8,
0x8d421fc0, 0x9b0ed10c, 0x88f1a1e9, 0x54c1f029,
0x7dead57b, 0x8d7ba426, 0x4cf5178a, 0x551a7cca,
0x1a9a5f08, 0xfcd651b9, 0x25605182, 0xe11fc6c3,
0xb6fd9676, 0x337b3027, 0xb7c8eb14, 0x9e5fd030,
0x6b57e354, 0xad913cf7, 0x7e16688d, 0x58872a69,
0x2c2fc7df, 0xe389ccc6, 0x30738df1, 0x0824a734,
0xe1797a8b, 0xa4a8d57b, 0x5b5d193b, 0xc8a8309b,
0x73f9a978, 0x73398d32, 0x0f59573e, 0xe9df2b03,
0xe8a5b6c8, 0x848d0704, 0x98df93c2, 0x720a1dc3,
0x684f259a, 0x943ba848, 0xa6370152, 0x863b5ea3,
0xd17b978b, 0x6d9b58ef, 0x0a700dd4, 0xa73d36bf,
0x8e6a0829, 0x8695bc14, 0xe35b3447, 0x933ac568,
0x8894b022, 0x2f511c27, 0xddfbcc3c, 0x006662b6,
0x117c83fe, 0x4e12b414, 0xc2bca766, 0x3a2fec10,
0xf4562420, 0x55792e2a, 0x46f5d857, 0xceda25ce,
0xc3601d3b, 0x6c00ab46, 0xefac9c28, 0xb3c35047,
0x611dfee3, 0x257c3207, 0xfdd58482, 0x3b14d84f,
0x23becb64, 0xa075f3a3, 0x088f8ead, 0x07adf158,
0x7796943c, 0xfacabf3d, 0xc09730cd, 0xf7679969,
0xda44e9ed, 0x2c854c12, 0x35935fa3, 0x2f057d9f,
0x690624f8, 0x1cb0bafd, 0x7b0dbdc6, 0x810f23bb,
0xfa929a1a, 0x6d969a17, 0x6742979b, 0x74ac7d05,
0x010e65c4, 0x86a3d963, 0xf907b5a0, 0xd0042bd3,
0x158d7d03, 0x287a8255, 0xbba8366f, 0x096edc33,
0x21916a7b, 0x77b56b86, 0x951622f9, 0xa6c5e650,
0x8cea17d1, 0xcd8c62bc, 0xa3d63433, 0x358a68fd,
0x0f9b9d3c, 0xd6aa295b, 0xfe33384a, 0xc000738e,
0xcd67eb2f, 0xe2eb6dc2, 0x97338b02, 0x06c9f246,
0x419cf1ad, 0x2b83c045, 0x3723f18a, 0xcb5b3089,
0x160bead7, 0x5d494656, 0x35f8a74b, 0x1e4e6c9e,
0x000399bd, 0x67466880, 0xb4174831, 0xacf423b2,
0xca815ab3, 0x5a6395e7, 0x302a67c5, 0x8bdb446b,
0x108f8fa4, 0x10223eda, 0x92b8b48b, 0x7f38d0ee,
0xab2701d4, 0x0262d415, 0xaf224a30, 0xb3d88aba,
0xf8b2c3af, 0xdaf7ef70, 0xcc97d3b7, 0xe9614b6c,
0x2baebff4, 0x70f687cf, 0x386c9156, 0xce092ee5,

```
    0x01e87da6, 0x6ce91e6a, 0xbb7bcc84, 0xc7922c20,
    0x9d3b71fd, 0x060e41c6, 0xd7590f15, 0x4e03bb47,
    0x183c198e, 0x63eeb240, 0x2ddbf49a, 0x6d5cba54,
    0x923750af, 0xf9e14236, 0x7838162b, 0x59726c72,
    0x81b66760, 0xbb2926c1, 0x48a0ce0d, 0xa6c0496d,
    0xad43507b, 0x718d496a, 0x9df057af, 0x44b1bde6,
    0x054356dc, 0xde7ced35, 0xd51a138b, 0x62088cc9,
    0x35830311, 0xc96efca2, 0x686f86ec, 0x8e77cb68,
    0x63e1d6b8, 0xc80f9778, 0x79c491fd, 0x1b4c67f2,
    0x72698d7d, 0x5e368c31, 0xf7d95e2e, 0xa1d3493f,
    0xdcd9433e, 0x896f1552, 0x4bc4ca7a, 0xa6d1baf4,
    0xa5a96dcc, 0x0bef8b46, 0xa169fda7, 0x74df40b7,
    0x4e208804, 0x9a756607, 0x038e87c8, 0x20211e44,
    0x8b7ad4bf, 0xc6403f35, 0x1848e36d, 0x80bdb038,
    0x1e62891c, 0x643d2107, 0xbf04d6f8, 0x21092c8c,
    0xf644f389, 0x0778404e, 0x7b78adb8, 0xa2c52d53,
    0x42157abe, 0xa2253e2e, 0x7bf3f4ae, 0x80f594f9,
    0x953194e7, 0x77eb92ed, 0xb3816930, 0xda8d9336,
    0xbf447469, 0xf26d9483, 0xee6faed5, 0x71371235,
    0xde425f73, 0xb4e59f43, 0x7dbe2d4e, 0x2d37b185,
    0x49dc9a63, 0x98c39d98, 0x1301c9a2, 0x389b1bbf,
    0x0c18588d, 0xa421c1ba, 0x7aa3865c, 0x71e08558,
    0x3c5cfcaa, 0x7d239ca4, 0x0297d9dd, 0xd7dc2830,
    0x4b37802b, 0x7428ab54, 0xaeee0347, 0x4b3fbb85,
    0x692f2f08, 0x134e578e, 0x36d9e0bf, 0xae8b5fcf,
    0xedb93ecf, 0x2b27248e, 0x170eb1ef, 0x7dc57fd6,
    0x1e760f16, 0xb1136601, 0x864e1b9b, 0xd7ea7319,
    0x3ab871bd, 0xcfa4d76f, 0xe31bd782, 0x0dbeb469,
    0xabb96061, 0x5370f85d, 0xffb07e37, 0xda30d0fb,
    0xebc977b6, 0x0b98b40f, 0x3a4d0fe6, 0xdf4fc26b,
    0x159cf22a, 0xc298d6e2, 0x2b78ef6a, 0x61a94ac0,
    0xab561187, 0x14eea0f0, 0xdf0d4164, 0x19af70ee
};

/*
 * The following implements Intermediate Values Tests Macros.
 * Since internal words are always kept little-endian, always
 * swap bytes before displaying.
 */
#ifdef IVT
int ivt_debug = 0;
FILE *ivt_fp;
int ivt_l = 0;
#define IVTSWAP(x) ¥
      ( (( x                ) << 24) | ¥
        (((x)&0xff00    ) << 8 ) | ¥
        (((x)&0xff0000) >> 8 ) | ¥
        (( x                ) >> 24)    )
#define IVT_DEBUG(a,b,c,d) ¥
      if (ivt_debug) ¥
          fprintf(ivt_fp,"IV%d=%8.8lx %8.8lx %8.8lx %8.8lx¥n",ivt_l++,¥
          IVTSWAP(a),IVTSWAP(b),IVTSWAP(c),IVTSWAP(d));
#else
#define IVT_DEBUG(a,b,c,d)
#endif

/*****************************************************************************
 *
 *     Low Level key setup, block encrypt and decrypt routines.
```

```
 *     For efficiency, these are WORD oriented. The high level NIST
 *     routines provide BYTE oriented interfaces, with ENDIAN conversion.
 *
 ************************************************************************/
/* if multiplication subkey k has 10 0's or 10 1's, mask in a fixing value */
static DWORD fix_subkey(DWORD k, DWORD r)
{
      /* the mask words come from S[265]..S[268], as chosen by index.c */
      DWORD *B = &S[265];
      DWORD m1, m2;
      int i;

      i = k & 3;                    /* store the least two bits of k */
      k |= 3;                       /* and then mask them away       */

      /* we look for 9 consequtive 1's in m1 */
      m1 = (~k) ^ (k<<1);     /* for i > 1, m1_i = 1 iff k_i = k_{i-1} */
      m2 = m1 & (m1 << 1);    /* m2_i = AND (m1_i, m1_{i-1})     */
      m2 &= m2 << 2;           /* m2_i = AND (m1_i...m1_{i-3})    */
      m2 &= m2 << 4;           /* m2_i = AND (m1_i...m1_{i-7})    */
      m2 &= m1 << 8;           /* m2_i = AND (m1_i...m1_{i-8})    */
      m2 &= 0xfffffe00;        /* mask out the low 9 bits of m2 */
      /* for i = 9...31, m2_i = 1 iff k_i = ... = k_{i-9} */

      /* if m2 is zero, k was good, so return */
      if (!m2)
            return(k);

      /* need to fix k: we copy each 1 in m2 to the nine bits to its right */
      m1 = m2 | (m2 >> 1);    /* m1_i = AND (m2_i, m2_{i+1})     */
      m1 |= m1 >> 2;           /* m1_i = AND (m2_i...m2_{i+3})    */
      m1 |= m2 >> 4;           /* m1_i = AND (m2_i...m2_{i+4})    */
      m1 |= m1 >> 5;           /* m1_i = AND (m2_i...m2_{i+9})    */
      /* m1_i = 1 iff k_i belongs to a sequence of ten 0's or ten 1's */

      /* we turn off the two lowest bits of M, and also every bit
       * M_i such that k_i is not equal to both k_{i-1} and k_{i+1}
       */
      m1 &= ((~k)^(k<<1)) & ((~k)^(k>>1)) & 0x7ffffffc;

      /* and finally pick a pattern, rotate it,
       * and xor it into k under the control of the mask m1
       */
       k ^= LROTATE(B[i], r) & m1;

       return(k);
}


/* setup a mars key schedule
 *
 * n    (input) is the number of words in the key
 * kp (input) is a pointer to the array of key words
 * ep (output) is a pointer to the array of EKEY_WORDS expanded subkey WORDs
 */
int mars_setup(int n, DWORD *kp, DWORD *ep)
{
      DWORD T[15] = {0};
      int i,j,t;
```

```
	/* check key length */
	if ((n<4)||(n>14))
		return(BAD_KEY_MAT);

	/* initialize the T[] array with key data */
	for (i=0; i<n; i++)
		T[i] = kp[i];
	T[n] = n;
	for (i=n+1; i<15; i++)
		T[i] = 0;

	/* Four iterations, each one computing 10 words of the array */
	for (j=0; j<4; j++) {
		DWORD w;

		/* Linear transformation */
		w = T[8] ^ T[13]; T[0] ^= LROTATE(w,3) ^ j;
		w = T[9] ^ T[14]; T[1] ^= LROTATE(w,3) ^ (4+j);
		for (i=2; i<7; i++) {
			w = T[i+8] ^ T[i-2];
			T[i] ^= LROTATE(w,3) ^ ((i<<2)+j);
		}
		for (i=7; i<15; i++) {
			w = T[i-7] ^ T[i-2];
			T[i] ^= LROTATE(w,3) ^ ((i<<2)+j);
		}

		/* Four stirring rounds */
		for (t=0; t<4; t++){
			/* stir with full type-1 s-box rounds */
			T[0] += S[ T[14]&511 ];
			T[0] = LROTATE(T[0],9);
			for (i=1; i<15; i++) {
				T[i] += S[ T[i-1]&511 ];
				T[i] = LROTATE(T[i],9);
			}
		}

		/* copy subkeys to mars_ctx, with swapping around */

#define SWAP(i) (ep[(10*j)+i] = T[(i*4)%15])

		SWAP(0); SWAP(1); SWAP(2); SWAP(3); SWAP(4);
		SWAP(5); SWAP(6); SWAP(7); SWAP(8); SWAP(9);
	}

	/* check and fix all multiplication subkeys */
	for (i=NUM_DATA+1;i<(EKEY_DWORDS-NUM_DATA);i+=2)
		ep[i] = fix_subkey(ep[i], ep[i-1]);

	return(TRUE);
}


#define MixForwardRound(d1,d2,d3,d4) ¥
	d2 ^= sp[d1&255];		¥
	y = RROTATE(d1,8);		¥
	z = RROTATE(d1,16);		¥
```

```
	d1 = RROTATE(d1,24);     ¥
	d2 += sp[(y&255)+256]; ¥
	d3 += sp[z&255];          ¥
	d4 ^= sp[(d1&255)+256];

#define MixBackwardsRound(d1,d2,d3,d4) ¥
	d2 ^= sp[(d1&255)+256];¥
	y = LROTATE(d1,8);        ¥
	z = LROTATE(d1,16);       ¥
	d1 = LROTATE(d1,24);     ¥
	d3 -= sp[y&255];          ¥
	d4 -= sp[(z&255)+256]; ¥
	d4 ^= sp[d1&255];

#define CoreRound(d1,d2,d3,d4,i) ¥
	y = d1;                   ¥
	d1 += ep[i];           ¥
	y = LROTATE(y,13); ¥
	z = d1;                   ¥
	tmp = y;                  ¥
	y *= ep[(i)+1];       ¥
	z &= 511;                 ¥
	z = sp[z];             ¥
	y = LROTATE(y,5);    ¥
	z ^= y;                   ¥
	d1 = LROTATE(d1,y);¥
	y = LROTATE(y,5);    ¥
	d3 += d1;                 ¥
	z ^= y;                   ¥
	z = LROTATE(z,y);    ¥
	d2 += z;                  ¥
	d4 ^= y;                  ¥
	d1 = tmp;

#define InvCoreRound(d1, d2, d3, d4, i) ¥
	y = d1;                    ¥
	d1 = RROTATE(d1,13);¥
	y *= ep[(i)+1];        ¥
	tmp = d1;                  ¥
	d1 += ep[i];            ¥
	z = d1;                   ¥
	y = LROTATE(y,5);     ¥
	z &= 511;                 ¥
	z = sp[z];             ¥
	d1 = LROTATE(d1,y); ¥
	z ^= y;                   ¥
	y = LROTATE(y,5);     ¥
	d3 -= d1;                 ¥
	z ^= y;                   ¥
	d1 = tmp;                  ¥
	z = LROTATE(z,y);     ¥
	d4 ^= y;                  ¥
	d2 -= z;

/* The basic mars encryption: (ep is the expanded key array) */
void mars_encrypt(DWORD *in, DWORD *out, DWORD *ep)
{
	register DWORD a,b,c,d,y,z;
	DWORD tmp;
```

```
    DWORD *sp = S;

    a = in[0];
    b = in[1];
    c = in[2];
    d = in[3];
    IVT_DEBUG(a,b,c,d);

#ifndef NO_MIX
    /* first, add subkeys to all input data words */
    a += ep[0];
    b += ep[1];
    c += ep[2];
    d += ep[3];
    IVT_DEBUG(a,b,c,d);

    /* then do eight mixing rounds */
    MixForwardRound(a,b,c,d);
    a += d;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(b,c,d,a);
    b += c;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(c,d,a,b);
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(d,a,b,c);
    IVT_DEBUG(a,b,c,d);

    MixForwardRound(a,b,c,d);
    a += d;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(b,c,d,a);
    b += c;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(c,d,a,b);
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(d,a,b,c);
    IVT_DEBUG(a,b,c,d);
#endif

    /* then sixteen mars encrypting rounds                    *
     * (eight in forward- and eight in backwards-mode) */

    CoreRound(a,b,c,d,4);
    IVT_DEBUG(a,b,c,d);
    CoreRound(b,c,d,a,6);
    IVT_DEBUG(a,b,c,d);
    CoreRound(c,d,a,b,8);
    IVT_DEBUG(a,b,c,d);
    CoreRound(d,a,b,c,10);
    IVT_DEBUG(a,b,c,d);

    CoreRound(a,b,c,d,12);
    IVT_DEBUG(a,b,c,d);
    CoreRound(b,c,d,a,14);
    IVT_DEBUG(a,b,c,d);
    CoreRound(c,d,a,b,16);
    IVT_DEBUG(a,b,c,d);
    CoreRound(d,a,b,c,18);
```

```
        IVT_DEBUG(a,b,c,d);

        CoreRound(a,d,c,b,20);
        IVT_DEBUG(a,b,c,d);
        CoreRound(b,a,d,c,22);
        IVT_DEBUG(a,b,c,d);
        CoreRound(c,b,a,d,24);
        IVT_DEBUG(a,b,c,d);
        CoreRound(d,c,b,a,26);
        IVT_DEBUG(a,b,c,d);

        CoreRound(a,d,c,b,28);
        IVT_DEBUG(a,b,c,d);
        CoreRound(b,a,d,c,30);
        IVT_DEBUG(a,b,c,d);
        CoreRound(c,b,a,d,32);
        IVT_DEBUG(a,b,c,d);
        CoreRound(d,c,b,a,34);
        IVT_DEBUG(a,b,c,d);

#ifndef NO_MIX
        /* then do eight inverse-mixing rounds */
        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(b,c,d,a);
        c -= b;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(c,d,a,b);
        d -= a;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(a,b,c,d);

        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(b,c,d,a);
        c -= b;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(c,d,a,b);
        d -= a;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(a,b,c,d);

        /* subtract final subkeys */
        a -= ep[2*NUM_ROUNDS+4];
        b -= ep[2*NUM_ROUNDS+5];
        c -= ep[2*NUM_ROUNDS+6];
        d -= ep[2*NUM_ROUNDS+7];
        IVT_DEBUG(a,b,c,d);
#endif

        out[0] = a;
        out[1] = b;
        out[2] = c;
        out[3] = d;
}

/* mars decryption is simply encryption in reverse */
```

```
void mars_decrypt(DWORD *in, DWORD *out, DWORD *ep)
{
	register DWORD a,b,c,d,y,z;
	DWORD tmp;
	DWORD *sp = S;

	d = in[0];
	c = in[1];
	b = in[2];
	a = in[3];
	IVT_DEBUG(d,c,b,a);

#ifndef NO_MIX
	/* first, add subkeys to all input data DWORDs */
	a += ep[2*NUM_ROUNDS+7];
	b += ep[2*NUM_ROUNDS+6];
	c += ep[2*NUM_ROUNDS+5];
	d += ep[2*NUM_ROUNDS+4];
	IVT_DEBUG(d,c,b,a);

	/* then do eight mixing rounds */
	MixForwardRound(a,b,c,d);
	IVT_DEBUG(d,c,b,a);
	a += d;
	MixForwardRound(b,c,d,a);
	IVT_DEBUG(d,c,b,a);
	b += c;
	MixForwardRound(c,d,a,b);
	IVT_DEBUG(d,c,b,a);
	MixForwardRound(d,a,b,c);
	IVT_DEBUG(d,c,b,a);

	MixForwardRound(a,b,c,d);
	IVT_DEBUG(d,c,b,a);
	a += d;
	MixForwardRound(b,c,d,a);
	IVT_DEBUG(d,c,b,a);
	b += c;
	MixForwardRound(c,d,a,b);
	IVT_DEBUG(d,c,b,a);
	MixForwardRound(d,a,b,c);
	IVT_DEBUG(d,c,b,a);
#endif

	/* then sixteen mars decrypting rounds                    *
	 * (eight in forward- and eight in backwards-mode) */

	InvCoreRound(a,b,c,d,34);
	IVT_DEBUG(d,c,b,a);
	InvCoreRound(b,c,d,a,32);
	IVT_DEBUG(d,c,b,a);
	InvCoreRound(c,d,a,b,30);
	IVT_DEBUG(d,c,b,a);
	InvCoreRound(d,a,b,c,28);
	IVT_DEBUG(d,c,b,a);

	InvCoreRound(a,b,c,d,26);
	IVT_DEBUG(d,c,b,a);
	InvCoreRound(b,c,d,a,24);
```

```
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(c,d,a,b,22);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(d,a,b,c,20);
        IVT_DEBUG(d,c,b,a);

        InvCoreRound(a,d,c,b,18);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(b,a,d,c,16);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(c,b,a,d,14);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(d,c,b,a,12);
        IVT_DEBUG(d,c,b,a);

        InvCoreRound(a,d,c,b,10);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(b,a,d,c,8);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(c,b,a,d,6);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(d,c,b,a,4);
        IVT_DEBUG(d,c,b,a);

#ifndef NO_MIX
        /* then do eight inverse-mixing rounds */
        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(d,c,b,a);
        MixBackwardsRound(b,c,d,a);
        IVT_DEBUG(d,c,b,a);
        c -= b;
        MixBackwardsRound(c,d,a,b);
        IVT_DEBUG(d,c,b,a);
        d -= a;
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(d,c,b,a);

        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(d,c,b,a);
        MixBackwardsRound(b,c,d,a);
        IVT_DEBUG(d,c,b,a);
        c -= b;
        MixBackwardsRound(c,d,a,b);
        IVT_DEBUG(d,c,b,a);
        d -= a;
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(d,c,b,a);

        /* subtract final subkeys */
        a -= ep[3];
        b -= ep[2];
        c -= ep[1];
        d -= ep[0];
        IVT_DEBUG(d,c,b,a);
#endif

        out[0] = d;
        out[1] = c;
        out[2] = b;
```

```
    out[3] = a;
}

/*****************************************************************************
 *
 *     NIST High Level key setup, block encrypt and decrypt routines
 *
 *****************************************************************************/

/* table for rapid, case insensitive hex conversion */
BYTE hex[256] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 0, 0, 0, 0, 0,
    0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

/* NIST defined high level key setup */
int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial)
{
    DWORD tmpkey[EKEY_DWORDS];
    int i,j;

    /* sanity check pointers */
    if (key == NULL)
        return(BAD_KEY_INSTANCE);
    if (keyMaterial == NULL)
        return(BAD_KEY_MAT);

    /* save parameters into keyInstance */
    key->direction = direction;
    key->keyLen = keyLen;
    for (i=0;i<keyLen/4;i++)
        key->keyMaterial[i] = keyMaterial[i];
    key->keyMaterial[MAX_KEY_SIZE] = '¥0';

    /* convert ascii keyMaterial to BYTEs */
    for(i=0,j=0;i<keyLen/4;i+=2,j++)
        ((BYTE *)tmpkey)[j] = (BYTE)((hex[(int)keyMaterial[i]]<<4) |
                                     hex[(int)keyMaterial[i+1]]);
#   ifdef SWAP_BYTES
        /* BSWAP the input key DWORDs */
        for (i=0;i<keyLen/W;i++)
            tmpkey[i] = BSWAP(tmpkey[i]);
#   endif

    /* call low level mars setup routine */
    return(mars_setup(keyLen/32,tmpkey,key->E));
```

```
}

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV)
{
    int i,j;

    /* sanity check pointers */
    if (cipher == NULL)
        return(BAD_CIPHER_MODE);

    /* save cipher parameters */
    cipher->mode = mode;

    /* handle IV */
    if((mode == MODE_CBC)||(mode == MODE_CFB1)) {
        if(IV == NULL)
            return(BAD_CIPHER_MODE);
        /* convert ascii IV to BYTEs */
        for(i=0,j=0;j<MAX_IV_SIZE;i+=2,j++)
            cipher->IV[j] = (BYTE)((hex[(int)IV[i]]<<4) | hex[(int)IV[i+1]]);
        /* copy BYTE IV to DWORD CIV, with conversion if necessary */
        for(i=0;i<NUM_DATA;i++)
            cipher->CIV[i] = BSWAP(((DWORD *)&cipher->IV)[i]);
    }
    return(TRUE);
}

/* this assumes the input length is a multiple of 128 bits */
int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer)
{
    DWORD tmp[4];
    int i;

    if (cipher->mode == MODE_ECB) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#           else
                mars_encrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
#           endif
        }
    }
    else if(cipher->mode == MODE_CBC) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                tmp[0] = BSWAP(*(DWORD *)(input+i+0)) ^ cipher->CIV[0];
                tmp[1] = BSWAP(*(DWORD *)(input+i+4)) ^ cipher->CIV[1];
                tmp[2] = BSWAP(*(DWORD *)(input+i+8)) ^ cipher->CIV[2];
                tmp[3] = BSWAP(*(DWORD *)(input+i+12)) ^ cipher->CIV[3];
                mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
```

```
                    cipher->CIV[0] = *(DWORD *)(outBuffer+i+0);
                    cipher->CIV[1] = *(DWORD *)(outBuffer+i+4);
                    cipher->CIV[2] = *(DWORD *)(outBuffer+i+8);
                    cipher->CIV[3] = *(DWORD *)(outBuffer+i+12);
                    *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                    *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                    *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                    *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#               else
                    tmp[0] = *(DWORD *)(input+i+0) ^ cipher->CIV[0];
                    tmp[1] = *(DWORD *)(input+i+4) ^ cipher->CIV[1];
                    tmp[2] = *(DWORD *)(input+i+8) ^ cipher->CIV[2];
                    tmp[3] = *(DWORD *)(input+i+12) ^ cipher->CIV[3];
                    mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                    cipher->CIV[0] = *(DWORD *)(outBuffer+i+0);
                    cipher->CIV[1] = *(DWORD *)(outBuffer+i+4);
                    cipher->CIV[2] = *(DWORD *)(outBuffer+i+8);
                    cipher->CIV[3] = *(DWORD *)(outBuffer+i+12);
#               endif
            }
        }
        else if(cipher->mode == MODE_CFB1) {
                    cipher->mode = MODE_ECB;        /* do encryption in ECB */
                    for (int n=0;n<inputLen;n++)
                            {
                            blockEncrypt(cipher,key,(BYTE       *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                            bit0    = 0x80 >> (n & 7);/* which bit position in byte */
                            ctBit = (input[n/8] & bit0) ^ ((((BYTE *) x)[0] & 0x80) >> (n&7));
                            outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) | ctBit;
                            carry = ctBit >> (7 - (n&7));
                            for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                    {
                                    bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                    cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                    carry = bit;
                                    }
                            }
                    cipher->mode = MODE_CFB1;       /* restore mode for next time */
                    return inputLen;
            /* Input is one bit (lsb of first byte).
              * Encrypt IV with key, xor input with msb of encrypted IV,
              * and then feed output cipher bit into lsb of IV.
              */
                    /*
            DWORD ECIV[4];

            if(inputLen != 1)
                return(BAD_CIPHER_MODE);

            mars_encrypt(cipher->CIV, ECIV, key->E);
            outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
            cipher->CIV[0] = (cipher->CIV[0]<<1) | (cipher->CIV[1] & 0x80000000);
            cipher->CIV[1] = (cipher->CIV[1]<<1) | (cipher->CIV[2] & 0x80000000);
            cipher->CIV[2] = (cipher->CIV[2]<<1) | (cipher->CIV[3] & 0x80000000);
            cipher->CIV[3] = (cipher->CIV[3]<<1) | (DWORD)outBuffer[0];*/
        }
        else
            return(BAD_CIPHER_MODE);
```

```
    return(inputLen);
}

/* this assumes the input length is a multiple of 128 bits */
int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer)
{
    int i;

    if (cipher->mode == MODE_ECB) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                DWORD tmp[4];
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_decrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#           else
                mars_decrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
#           endif
        }
    }
    else if(cipher->mode == MODE_CBC) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                DWORD tmp[4];
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_decrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0)
                    ^ cipher->CIV[0]);
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4)
                    ^ cipher->CIV[1]);
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8)
                    ^ cipher->CIV[2]);
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12)
                    ^ cipher->CIV[3]);
                cipher->CIV[0] = tmp[0];
                cipher->CIV[1] = tmp[1];
                cipher->CIV[2] = tmp[2];
                cipher->CIV[3] = tmp[3];
#           else
                mars_decrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) ^= cipher->CIV[0];
                *(DWORD *)(outBuffer+i+4) ^= cipher->CIV[1];
                *(DWORD *)(outBuffer+i+8) ^= cipher->CIV[2];
                *(DWORD *)(outBuffer+i+12) ^= cipher->CIV[3];
                cipher->CIV[0] = *(DWORD *)(input+i+0);
                cipher->CIV[1] = *(DWORD *)(input+i+4);
                cipher->CIV[2] = *(DWORD *)(input+i+8);
                cipher->CIV[3] = *(DWORD *)(input+i+12);
```

```
#               endif
        }
    }
    else if(cipher->mode == MODE_CFB1){
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,(BYTE       *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                        bit0    = 0x80 >> (n & 7);
                        ctBit = input[n/8] & bit0;
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) |
                                                        (ctBit ^ ((((BYTE *) x)[0] & 0x80)
>> (n&7)));
                        carry = ctBit >> (7 - (n&7));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;       /* restore mode for next time */
                return inputLen;


        /* Input is one bit (lsb of first byte).
          * Encrypt IV with key, xor input with msb of encrypted IV,
          * and then feed input cipher bit into lsb of IV.
          */
                /*
        DWORD ECIV[4];

        if(inputLen != 1)
            return(BAD_CIPHER_MODE);

        mars_encrypt(cipher->CIV, ECIV, key->E);
        outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
        cipher->CIV[0] = (cipher->CIV[0]<<1) | (cipher->CIV[1] & 0x80000000);
        cipher->CIV[1] = (cipher->CIV[1]<<1) | (cipher->CIV[2] & 0x80000000);
        cipher->CIV[2] = (cipher->CIV[2]<<1) | (cipher->CIV[3] & 0x80000000);
        cipher->CIV[3] = (cipher->CIV[3]<<1) | (DWORD)(input[0]&1);
*/
    }
    else
        return(BAD_CIPHER_MODE);

    return(inputLen);
}


//////////////////////////////////////////////////////////

// SerpentEC.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include <stdio.h>
```

```
#include <time.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <malloc.h>
#include "aes.h"


#define file_len(x) (unsigned long)x

int s;
keyInstance keyI;
cipherInstance cipherI;


int main(int argc, char* argv[])
{
	FILE *fkey =0, *fin = 0, *fout = 0;
    fpos_t flen;
    unsigned long len, rlen, blen4;
	int mode,klen,blen,rc;
	char c_mode[4], c_klen[8], c_key[128], c_cini[64];
	unsigned char pbuf[2048], cbuf[2048];

	blen4 = 2048;

	printf( "Start!¥n" );

	if( argc != 4 ){                                    // 使い方の誤り
		printf( "argc != 4 ¥n" );
		return -1;
	}

	if((fkey = fopen(argv[1], "rt")) == NULL){
		printf("Can not find key file for encryption. ¥n");
		return (-1);
	}

	fgets( c_mode, 4, fkey );
	fgets( c_klen, 8, fkey );
	fgets( c_key, 128, fkey );
	fgets( c_cini, 64, fkey );

	mode = atoi(c_mode);
	klen = atoi(c_klen);
	blen = 128;

	if(klen<56 || 448<klen){
		printf("Wrong key size. ¥n");
		return (-1);
	}

	/*Set mode*/
	if(mode == 1){
		rc=cipherInit(&cipherI, MODE_ECB, "");
	}
	if(mode == 2){
		rc=cipherInit(&cipherI, MODE_CBC, c_cini);
	}
```

```
        if(mode == 3){
                rc=cipherInit(&cipherI, MODE_CFB1, c_cini);
        }
        if(rc<=0){
                printf("モード設定が出来ません。");
                return(-2);
        }

        makeKey(&keyI, DIR_ENCRYPT,    klen, c_key );

        if((fin = fopen(argv[2], "rb")) == NULL){
                printf("Can not open plane text file. ¥n");
                return (-1);
        }

    fseek(fin, 0, SEEK_END);
    fgetpos(fin, &flen);
    rlen = file_len(flen);
    // reset to start
    fseek(fin, 0, SEEK_SET);

        if((fout = fopen(argv[3], "wb")) == NULL){
                printf("Can not open encrypted data file. ¥n");
                return (-1);
        }

        s = sizeof(unsigned int);
        // write the bytes of the file
        *((unsigned int*)pbuf) = rlen;

        len = (unsigned long) fread(pbuf+s, 1, blen4-s , fin);
    rlen -= len;
        rc=blockEncrypt(&cipherI,          &keyI,(unsigned          char*)(pbuf),8*blen4,(unsigned
char*)(cbuf));
    if(fwrite(cbuf, 1, blen4, fout) != (unsigned int)blen4)
        return -2;

    while(rlen > 0 && !feof(fin)){
        // read a block and reduce the remaining byte count
        len = (unsigned long)fread(pbuf, 1, blen4 , fin);
        rlen -= len;
                rc=blockEncrypt(&cipherI,    &keyI,(unsigned    char*)(pbuf),8*blen4,(unsigned
char*)(cbuf));
        if(fwrite(cbuf, 1, blen4, fout) != blen4){return -2;}
    }

        if(fkey){ fclose(fkey);        }
        if(fout){ fclose(fout);        }
        if(fin)    { fclose(fin);        }
        printf( "End!¥n" );
        return 0;
}

////////////////////////////////////////////////////////////////////////////

// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//                  MarsEC.pch 生成されるプリコンパイル済ヘッダー
//                  stdafx.obj 生成されるプリコンパイル済タイプ情報
```

```
#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。

//////////////////////////////////////////////////////////////////

/*  aes.h  */

/*  AES Cipher header file for ANSI C Submissions
 *      Lawrence E. Bassham III
 *      Computer Security Division
 *      National Institute of Standards and Technology
 *
 *      April 15, 1998
 *
 *      Modified for IBM submission
 *      David Safford
 *      4/16/1998
 */

#include <stdio.h>

/*****************************************************************
 *
 *      NIST High Level C API, with some IBM additions
 *
 *****************************************************************/

#define TRUE                1
#define FALSE               0

#define DIR_ENCRYPT         0     /*  Are we encrypting?                          */
#define DIR_DECRYPT         1     /*  Are we decrypting?                          */
#define MODE_ECB            1     /*  Are we ciphering in ECB mode?               */
#define MODE_CBC            2     /*  Are we ciphering in CBC mode?               */
#define MODE_CFB1           3     /*  Are we ciphering in 1-bit CFB mode?         */

#define BAD_KEY_DIR         -1    /*  Key direction is invalid                    */
#define BAD_KEY_MAT         -2    /*  Key material not of correct length          */
#define BAD_KEY_INSTANCE -3   /*  Key passed is not valid                     */
#define BAD_CIPHER_MODE   -4   /*  Params struct passed to cipherInit invalid */
#define BAD_CIPHER_STATE -5   /*  Cipher in wrong state                       */

typedef unsigned char BYTE;

/* IBM Addition - a WORD must be 32 bits for this implementation */
typedef unsigned long DWORD;

/* IBM specific defines: these parameters can be changed                    */
#define NUM_MIX 8               /* number of mixing rounds per stage        */
#define NUM_ROUNDS 16           /* number of full core rounds               */
#define NUM_SETUP 7             /* number of key setup mixing rounds        */

/* IBM specific defines: these parameters are fixed for this implementation */
#define W               32          /* number of bits in a word             */
#define NUM_DATA        4           /* data block size in words             */
#define  EKEY_DWORDS  (2*(NUM_DATA+NUM_ROUNDS))     /*  number  of  subkey  words
*/
```

```
/* IBM modified values                                                    */
#define MAX_KEY_SIZE (EKEY_DWORDS*8)   /* max ASCII char's needed for a key     */
#define MAX_IV_SIZE (NUM_DATA*4)       /* max bytes's needed for an IV          */

/*   The structure for key information */
typedef struct {
    BYTE    direction;          /*   Key used for encrypting or decrypting?          */
    int     keyLen;             /*   Length of the key in BITS                       */
    char    keyMaterial[MAX_KEY_SIZE+1];   /*   Raw key data in ASCII                */
    DWORD   E[EKEY_DWORDS];     /* IBM addition for mars expanded key                */
} keyInstance;

/*   The structure for cipher information */
typedef struct {
    BYTE mode;                  /*   MODE_ECB, MODE_CBC, or MODE_CFB1              */
    BYTE IV[MAX_IV_SIZE];   /*   initial binary IV BYTE for chaining               */
    DWORD CIV[NUM_DATA];        /*   IBM addition: current IV in binary WORDs      */
} cipherInstance;

/*   NIST High level function protoypes   */
int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial);

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer);

int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer);

/*****************************************************************************
 *
 *      IBM Low Level (WORD Oriented) API
 *
 *****************************************************************************/

/* setup a mars expanded key
 *
 * k   (input) is the number of words in the key
 * kp (input) is a pointer to the array of k key words
 * ep (output) is a pointer to an array of EKEY_WORDS expanded subkey WORDs
 */
int mars_setup(int k, DWORD *kp, DWORD *ep);

/* The basic mars encryption of one block (of NUM_DATA WORDS) */
void mars_encrypt(DWORD *in, DWORD *out, DWORD *ep);

/* mars decryption is simply encryption in reverse */
void mars_decrypt(DWORD *in, DWORD *out, DWORD *ep);

////////////////////////////////////////////////////////////////////////////

// stdafx.h : 標準のシステム インクルード ファイル、
//            または参照回数が多く、かつあまり変更されない
//            プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__FCFDF2CE_CA62_4C73_B418_837F237E0404__INCLUDED_)
```

```
#define AFX_STDAFX_H__FCFDF2CE_CA62_4C73_B418_837F237E0404__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif
// !defined(AFX_STDAFX_H__FCFDF2CE_CA62_4C73_B418_837F237E0404__INCLUDED_)
```

１０．５．２　復号化部分

```
/* mars.c optimized C code - copyright(c) 1998 IBM
 *
 * This code implements both the NIST high level C API (version 5)
 * and an underlying IBM defined low level (DWORD oriented) C API.
 */

/* Revisions log:
 *
 *     Aug 1999, Shai - the "tweaked" key schedule
 *     Apr 1998, Dave && Shai - new key scheduling, new sbox, NIST API
 *     Mar 1998, Shai Halevi   - adapted to the latest variant of mars
 *     Feb 1998, Dave Safford - created
 */

/* Compilation using pcg's version of gcc:
 *       gcc -Wall -pedantic -c -O6 -fomit-frame-pointer -mcpu=pentiumpro
 *             -DINTEL_GCC tests.c mars-opt.c
 *
 * Compilation using xlc on AIX:
 *       xlc -c -O3 -DAIX_XLC mars-opt.c
 *
 * Compilation using Borland C++ 5.0 from a DOS command line:
 *       bcc32 -Oi -6 -v -A -a4 -O2 -DKAT tests.c mars-opt.c
 *
 * Useful compilation defines:
 *       NO_MIX         - do not execute the mixing phases (core only)
 *       INTEL_GCC    - optimized for pentiumpro and djgpp/gcc compiler
 *       AIX_XLC       - optimized for powerpc/AIX with xlc compiler
 *       SWAP_BYTES - force endian conversion (eg __BYTE_ORDER not supported)
 *       IVT              - add intermediate values test outputs (this slows
 *                          down encryption and decryption significantly)
 */
  #include "stdafx.h"
#include "aes.h"

/* The low level mars routines are completely DWORD oriented, and
```

```
 * endian neutral. The high level NIST routines provide BYTE oriented
 * inputs and outputs, thus raising the endian issue when converting
 * between BYTEs and DWORDs. For these conversions, mars assumes
 * little endian order. On a big endian machine, we define BSWAP()
 * to do the conversions. BSWAP() is about the best you can do in C.
 * Real implementations will undoubtedly use inline ASM, as most risc
 * machines can do this in one instruction.
 *
 * Make a best guess on platform endianness; This works on linux, AIX,
 * and W95. For other platforms, you may have to manually define SWAP_BYTES
 * on a big endian machine if this guessing doesn't work.
 */

#ifdef _AIX
#define SWAP_BYTES
#else
#ifdef __linux__
#include <endian.h>
#ifdef __BYTE_ORDER
#if __BYTE_ORDER == __BIG_ENDIAN
#define SWAP_BYTES
#endif
#endif
#endif
#endif
#ifdef SWAP_BYTES
#define BSWAP(x) ¥
      ( (( x                ) << 24) | ¥
        (((x)&0xff00    ) << 8 ) | ¥
        (((x)&0xff0000) >> 8 ) | ¥
        (( x                ) >> 24)    )
#else
#define BSWAP(x) (x)
#endif

/* Some compiler optimizers will recognize certain rotation idioms,
 * and reduce them to native rotation instructions. These idioms
 * have been found to work for GCC on Intel, and XLC on RS6000.
 * Nothing seems to work on Borland C++ 5.0, although we can leave
 * out the mask on 'b'.
 */
#ifdef INTEL_GCC
#define LROTATE(a,b) (((a)<<(int)(b)) | ((a)>>(W - (int)(b))))
#define RROTATE(a,b) (((a)>>(int)(b)) | ((a)<<(W - (int)(b))))
#else
#ifdef AIX_XLC
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#ifdef __BORLANDC__
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#define LROTATE(a,b) (((a)<<(int)(b&31)) | ((a)>>(W - (int)(b&31))))
#define RROTATE(a,b) (((a)>>(int)(b&31)) | ((a)<<(W - (int)(b&31))))
#endif
#endif
#endif
```

```
//////////////////////////////////////////
#define BLOCK_SIZE 128
   unsigned long t[4];
   int    b, n, i;
   DWORD x[BLOCK_SIZE/32];
   BYTE bit, bit0, ctBit, carry;
/* two 8 x 32 sboxes - stored together to save a pointer
  * these sboxes were generated with buf[3] = 0x02917d59, as
  * chosen by sbox.c, which had the following output:
  *     After 54817140, (38023 fail) new min j = 43089241 (0x2917d59)
  *     test 0 eval 0.044922 (single bit correlation)
  *     test 1 eval 0.033203 (single bit bias)
  *     test 2 eval 0.031250 (consecutive bit bias)
  *     test 3 eval 0.007813 (parity bias)
  *     test 4 eval 0.148438 (avalanche)
  */
static DWORD S[512] = {
   0x09d0c479, 0x28c8ffe0, 0x84aa6c39, 0x9dad7287,
   0x7dff9be3, 0xd4268361, 0xc96da1d4, 0x7974cc93,
   0x85d0582e, 0x2a4b5705, 0x1ca16a62, 0xc3bd279d,
   0x0f1f25e5, 0x5160372f, 0xc695c1fb, 0x4d7ff1e4,
   0xae5f6bf4, 0x0d72ee46, 0xff23de8a, 0xb1cf8e83,
   0xf14902e2, 0x3e981e42, 0x8bf53eb6, 0x7f4bf8ac,
   0x83631f83, 0x25970205, 0x76afe784, 0x3a7931d4,
   0x4f846450, 0x5c64c3f6, 0x210a5f18, 0xc6986a26,
   0x28f4e826, 0x3a60a81c, 0xd340a664, 0x7ea820c4,
   0x526687c5, 0x7eddd12b, 0x32a11d1d, 0x9c9ef086,
   0x80f6e831, 0xab6f04ad, 0x56fb9b53, 0x8b2e095c,
   0xb68556ae, 0xd2250b0d, 0x294a7721, 0xe21fb253,
   0xae136749, 0xe82aae86, 0x93365104, 0x99404a66,
   0x78a784dc, 0xb69ba84b, 0x04046793, 0x23db5c1e,
   0x46cae1d6, 0x2fe28134, 0x5a223942, 0x1863cd5b,
   0xc190c6e3, 0x07dfb846, 0x6eb88816, 0x2d0dcc4a,
   0xa4ccae59, 0x3798670d, 0xcbfa9493, 0x4f481d45,
   0xeafc8ca8, 0xdb1129d6, 0xb0449e20, 0x0f5407fb,
   0x6167d9a8, 0xd1f45763, 0x4daa96c3, 0x3bec5958,
   0xababa014, 0xb6ccd201, 0x38d6279f, 0x02682215,
   0x8f376cd5, 0x092c237e, 0xbfc56593, 0x32889d2c,
   0x854b3e95, 0x05bb9b43, 0x7dcd5dcd, 0xa02e926c,
   0xfae527e5, 0x36a1c330, 0x3412e1ae, 0xf257f462,
   0x3c4f1d71, 0x30a2e809, 0x68e5f551, 0x9c61ba44,
   0x5ded0ab8, 0x75ce09c8, 0x9654f93e, 0x698c0cca,
   0x243cb3e4, 0x2b062b97, 0x0f3b8d9e, 0x00e050df,
   0xfc5d6166, 0xe35f9288, 0xc079550d, 0x0591aee8,
   0x8e531e74, 0x75fe3578, 0x2f6d829a, 0xf60b21ae,
   0x95e8eb8d, 0x6699486b, 0x901d7d9b, 0xfd6d6e31,
   0x1090acef, 0xe0670dd8, 0xdab2e692, 0xcd6d4365,
   0xe5393514, 0x3af345f0, 0x6241fc4d, 0x460da3a3,
   0x7bcf3729, 0x8bf1d1e0, 0x14aac070, 0x1587ed55,
   0x3afd7d3e, 0xd2f29e01, 0x29a9d1f6, 0xefb10c53,
   0xcf3b870f, 0xb414935c, 0x664465ed, 0x024acac7,
   0x59a744c1, 0x1d2936a7, 0xdc580aa6, 0xcf574ca8,
   0x040a7a10, 0x6cd81807, 0x8a98be4c, 0xaccea063,
   0xc33e92b5, 0xd1e0e03d, 0xb322517e, 0x2092bd13,
   0x386b2c4a, 0x52e8dd58, 0x58656dfb, 0x50820371,
   0x41811896, 0xe337ef7e, 0xd39fb119, 0xc97f0df6,
   0x68fea01b, 0xa150a6e5, 0x55258962, 0xeb6ff41b,
   0xd7c9cd7a, 0xa619cd9e, 0xbcf09576, 0x2672c073,
   0xf003fb3c, 0x4ab7a50b, 0x1484126a, 0x487ba9b1,
```

0xa64fc9c6, 0xf6957d49, 0x38b06a75, 0xdd805fcd,
0x63d094cf, 0xf51c999e, 0x1aa4d343, 0xb8495294,
0xce9f8e99, 0xbffcd770, 0xc7c275cc, 0x378453a7,
0x7b21be33, 0x397f41bd, 0x4e94d131, 0x92cc1f98,
0x5915ea51, 0x99f861b7, 0xc9980a88, 0x1d74fd5f,
0xb0a495f8, 0x614deed0, 0xb5778eea, 0x5941792d,
0xfa90c1f8, 0x33f824b4, 0xc4965372, 0x3ff6d550,
0x4ca5fec0, 0x8630e964, 0x5b3fbbd6, 0x7da26a48,
0xb203231a, 0x04297514, 0x2d639306, 0x2eb13149,
0x16a45272, 0x532459a0, 0x8e5f4872, 0xf966c7d9,
0x07128dc0, 0x0d44db62, 0xafc8d52d, 0x06316131,
0xd838e7ce, 0x1bc41d00, 0x3a2e8c0f, 0xea83837e,
0xb984737d, 0x13ba4891, 0xc4f8b949, 0xa6d6acb3,
0xa215cdce, 0x8359838b, 0x6bd1aa31, 0xf579dd52,
0x21b93f93, 0xf5176781, 0x187dfdde, 0xe94aeb76,
0x2b38fd54, 0x431de1da, 0xab394825, 0x9ad3048f,
0xdfea32aa, 0x659473e3, 0x623f7863, 0xf3346c59,
0xab3ab685, 0x3346a90b, 0x6b56443e, 0xc6de01f8,
0x8d421fc0, 0x9b0ed10c, 0x88f1a1e9, 0x54c1f029,
0x7dead57b, 0x8d7ba426, 0x4cf5178a, 0x551a7cca,
0x1a9a5f08, 0xfcd651b9, 0x25605182, 0xe11fc6c3,
0xb6fd9676, 0x337b3027, 0xb7c8eb14, 0x9e5fd030,
0x6b57e354, 0xad913cf7, 0x7e16688d, 0x58872a69,
0x2c2fc7df, 0xe389ccc6, 0x30738df1, 0x0824a734,
0xe1797a8b, 0xa4a8d57b, 0x5b5d193b, 0xc8a8309b,
0x73f9a978, 0x73398d32, 0x0f59573e, 0xe9df2b03,
0xe8a5b6c8, 0x848d0704, 0x98df93c2, 0x720a1dc3,
0x684f259a, 0x943ba848, 0xa6370152, 0x863b5ea3,
0xd17b978b, 0x6d9b58ef, 0x0a700dd4, 0xa73d36bf,
0x8e6a0829, 0x8695bc14, 0xe35b3447, 0x933ac568,
0x8894b022, 0x2f511c27, 0xddfbcc3c, 0x006662b6,
0x117c83fe, 0x4e12b414, 0xc2bca766, 0x3a2fec10,
0xf4562420, 0x55792e2a, 0x46f5d857, 0xceda25ce,
0xc3601d3b, 0x6c00ab46, 0xefac9c28, 0xb3c35047,
0x611dfee3, 0x257c3207, 0xfdd58482, 0x3b14d84f,
0x23becb64, 0xa075f3a3, 0x088f8ead, 0x07adf158,
0x7796943c, 0xfacabf3d, 0xc09730cd, 0xf7679969,
0xda44e9ed, 0x2c854c12, 0x35935fa3, 0x2f057d9f,
0x690624f8, 0x1cb0bafd, 0x7b0dbdc6, 0x810f23bb,
0xfa929a1a, 0x6d969a17, 0x6742979b, 0x74ac7d05,
0x010e65c4, 0x86a3d963, 0xf907b5a0, 0xd0042bd3,
0x158d7d03, 0x287a8255, 0xbba8366f, 0x096edc33,
0x21916a7b, 0x77b56b86, 0x951622f9, 0xa6c5e650,
0x8cea17d1, 0xcd8c62bc, 0xa3d63433, 0x358a68fd,
0x0f9b9d3c, 0xd6aa295b, 0xfe33384a, 0xc000738e,
0xcd67eb2f, 0xe2eb6dc2, 0x97338b02, 0x06c9f246,
0x419cf1ad, 0x2b83c045, 0x3723f18a, 0xcb5b3089,
0x160bead7, 0x5d494656, 0x35f8a74b, 0x1e4e6c9e,
0x000399bd, 0x67466880, 0xb4174831, 0xacf423b2,
0xca815ab3, 0x5a6395e7, 0x302a67c5, 0x8bdb446b,
0x108f8fa4, 0x10223eda, 0x92b8b48b, 0x7f38d0ee,
0xab2701d4, 0x0262d415, 0xaf224a30, 0xb3d88aba,
0xf8b2c3af, 0xdaf7ef70, 0xcc97d3b7, 0xe9614b6c,
0x2baebff4, 0x70f687cf, 0x386c9156, 0xce092ee5,
0x01e87da6, 0x6ce91e6a, 0xbb7bcc84, 0xc7922c20,
0x9d3b71fd, 0x060e41c6, 0xd7590f15, 0x4e03bb47,
0x183c198e, 0x63eeb240, 0x2ddbf49a, 0x6d5cba54,
0x923750af, 0xf9e14236, 0x7838162b, 0x59726c72,
0x81b66760, 0xbb2926c1, 0x48a0ce0d, 0xa6c0496d,

```
    0xad43507b, 0x718d496a, 0x9df057af, 0x44b1bde6,
    0x054356dc, 0xde7ced35, 0xd51a138b, 0x62088cc9,
    0x35830311, 0xc96efca2, 0x686f86ec, 0x8e77cb68,
    0x63e1d6b8, 0xc80f9778, 0x79c491fd, 0x1b4c67f2,
    0x72698d7d, 0x5e368c31, 0xf7d95e2e, 0xa1d3493f,
    0xdcd9433e, 0x896f1552, 0x4bc4ca7a, 0xa6d1baf4,
    0xa5a96dcc, 0x0bef8b46, 0xa169fda7, 0x74df40b7,
    0x4e208804, 0x9a756607, 0x038e87c8, 0x20211e44,
    0x8b7ad4bf, 0xc6403f35, 0x1848e36d, 0x80bdb038,
    0x1e62891c, 0x643d2107, 0xbf04d6f8, 0x21092c8c,
    0xf644f389, 0x0778404e, 0x7b78adb8, 0xa2c52d53,
    0x42157abe, 0xa2253e2e, 0x7bf3f4ae, 0x80f594f9,
    0x953194e7, 0x77eb92ed, 0xb3816930, 0xda8d9336,
    0xbf447469, 0xf26d9483, 0xee6faed5, 0x71371235,
    0xde425f73, 0xb4e59f43, 0x7dbe2d4e, 0x2d37b185,
    0x49dc9a63, 0x98c39d98, 0x1301c9a2, 0x389b1bbf,
    0x0c18588d, 0xa421c1ba, 0x7aa3865c, 0x71e08558,
    0x3c5cfcaa, 0x7d239ca4, 0x0297d9dd, 0xd7dc2830,
    0x4b37802b, 0x7428ab54, 0xaeee0347, 0x4b3fbb85,
    0x692f2f08, 0x134e578e, 0x36d9e0bf, 0xae8b5fcf,
    0xedb93ecf, 0x2b27248e, 0x170eb1ef, 0x7dc57fd6,
    0x1e760f16, 0xb1136601, 0x864e1b9b, 0xd7ea7319,
    0x3ab871bd, 0xcfa4d76f, 0xe31bd782, 0x0dbeb469,
    0xabb96061, 0x5370f85d, 0xffb07e37, 0xda30d0fb,
    0xebc977b6, 0x0b98b40f, 0x3a4d0fe6, 0xdf4fc26b,
    0x159cf22a, 0xc298d6e2, 0x2b78ef6a, 0x61a94ac0,
    0xab561187, 0x14eea0f0, 0xdf0d4164, 0x19af70ee
};

/*
 * The following implements Intermediate Values Tests Macros.
 * Since internal words are always kept little-endian, always
 * swap bytes before displaying.
 */
#ifdef IVT
int ivt_debug = 0;
FILE *ivt_fp;
int ivt_l = 0;
#define IVTSWAP(x) ¥
      ( (( x                ) << 24) | ¥
        (((x)&0xff00    ) << 8 ) | ¥
        (((x)&0xff0000) >> 8 ) | ¥
        (( x                ) >> 24)    )
#define IVT_DEBUG(a,b,c,d) ¥
      if (ivt_debug) ¥
           fprintf(ivt_fp,"IV%d=%8.8lx %8.8lx %8.8lx %8.8lx¥n",ivt_l++,¥
           IVTSWAP(a),IVTSWAP(b),IVTSWAP(c),IVTSWAP(d));
#else
#define IVT_DEBUG(a,b,c,d)
#endif

/*****************************************************************************
 *
 *     Low Level key setup, block encrypt and decrypt routines.
 *     For efficiency, these are WORD oriented. The high level NIST
 *     routines provide BYTE oriented interfaces, with ENDIAN conversion.
 *
 *****************************************************************************/
/* if multiplication subkey k has 10 0's or 10 1's, mask in a fixing value */
```

```
static DWORD fix_subkey(DWORD k, DWORD r)
{
	/* the mask words come from S[265]..S[268], as chosen by index.c */
	DWORD *B = &S[265];
	DWORD m1, m2;
	int i;

	i = k & 3;			/* store the least two bits of k */
	k |= 3;				 /* and then mask them away		*/

	/* we look for 9 consequtive 1's in m1 */
	m1 = (~k) ^ (k<<1);	/* for i > 1, m1_i = 1 iff k_i = k_{i-1} */
	m2 = m1 & (m1 << 1);	/* m2_i = AND (m1_i, m1_{i-1})	*/
	m2 &= m2 << 2;		 /* m2_i = AND (m1_i...m1_{i-3})	*/
	m2 &= m2 << 4;		 /* m2_i = AND (m1_i...m1_{i-7})	*/
	m2 &= m1 << 8;		 /* m2_i = AND (m1_i...m1_{i-8})	*/
	m2 &= 0xfffffe00;	/* mask out the low 9 bits of m2 */
	/* for i = 9...31, m2_i = 1 iff k_i = ... = k_{i-9} */

	/* if m2 is zero, k was good, so return */
	if (!m2)
		return(k);

	/* need to fix k: we copy each 1 in m2 to the nine bits to its right */
	m1 = m2 | (m2 >> 1);	/* m1_i = AND (m2_i, m2_{i+1})	*/
	m1 |= m1 >> 2;		 /* m1_i = AND (m2_i...m2_{i+3})	*/
	m1 |= m2 >> 4;		 /* m1_i = AND (m2_i...m2_{i+4})	*/
	m1 |= m1 >> 5;		 /* m1_i = AND (m2_i...m2_{i+9})	*/
	/* m1_i = 1 iff k_i belongs to a sequence of ten 0's or ten 1's */

	/* we turn off the two lowest bits of M, and also every bit
	 * M_i such that k_i is not equal to both k_{i-1} and k_{i+1}
	 */
	m1 &= ((~k)^(k<<1)) & ((~k)^(k>>1)) & 0x7ffffffc;

	/* and finally pick a pattern, rotate it,
	 * and xor it into k under the control of the mask m1
	 */
	 k ^= LROTATE(B[i], r) & m1;

	 return(k);
}


/* setup a mars key schedule
 *
 * n	(input) is the number of words in the key
 * kp (input) is a pointer to the array of key words
 * ep (output) is a pointer to the array of EKEY_WORDS expanded subkey WORDs
 */
int mars_setup(int n, DWORD *kp, DWORD *ep)
{
	DWORD T[15] = {0};
	int i,j,t;

	/* check key length */
	if ((n<4)||(n>14))
		return(BAD_KEY_MAT);
```

```
        /* initialize the T[] array with key data */
        for (i=0; i<n; i++)
                T[i] = kp[i];
        T[n] = n;
        for (i=n+1; i<15; i++)
                T[i] = 0;

        /* Four iterations, each one computing 10 words of the array */
        for (j=0; j<4; j++) {
                DWORD w;

                /* Linear transformation */
                w = T[8] ^ T[13]; T[0] ^= LROTATE(w,3) ^ j;
                w = T[9] ^ T[14]; T[1] ^= LROTATE(w,3) ^ (4+j);
                for (i=2; i<7; i++) {
                        w = T[i+8] ^ T[i-2];
                        T[i] ^= LROTATE(w,3) ^ ((i<<2)+j);
                }
                for (i=7; i<15; i++) {
                        w = T[i-7] ^ T[i-2];
                        T[i] ^= LROTATE(w,3) ^ ((i<<2)+j);
                }

                /* Four stirring rounds */
                for (t=0; t<4; t++){
                        /* stir with full type-1 s-box rounds */
                        T[0] += S[ T[14]&511 ];
                        T[0] = LROTATE(T[0],9);
                        for (i=1; i<15; i++) {
                                T[i] += S[ T[i-1]&511 ];
                                T[i] = LROTATE(T[i],9);
                        }
                }

                /* copy subkeys to mars_ctx, with swapping around */

#define SWAP(i) (ep[(10*j)+i] = T[(i*4)%15])

                SWAP(0); SWAP(1); SWAP(2); SWAP(3); SWAP(4);
                SWAP(5); SWAP(6); SWAP(7); SWAP(8); SWAP(9);
        }

        /* check and fix all multiplication subkeys */
        for (i=NUM_DATA+1;i<(EKEY_DWORDS-NUM_DATA);i+=2)
                ep[i] = fix_subkey(ep[i], ep[i-1]);

        return(TRUE);
}


#define MixForwardRound(d1,d2,d3,d4) ¥
        d2 ^= sp[d1&255];          ¥
        y = RROTATE(d1,8);         ¥
        z = RROTATE(d1,16);        ¥
        d1 = RROTATE(d1,24);       ¥
        d2 += sp[(y&255)+256]; ¥
        d3 += sp[z&255];           ¥
        d4 ^= sp[(d1&255)+256];
```

```
#define MixBackwardsRound(d1,d2,d3,d4) ¥
	d2 ^= sp[(d1&255)+256];¥
	y = LROTATE(d1,8);	¥
	z = LROTATE(d1,16);	¥
	d1 = LROTATE(d1,24);	¥
	d3 -= sp[y&255];	¥
	d4 -= sp[(z&255)+256]; ¥
	d4 ^= sp[d1&255];

#define CoreRound(d1,d2,d3,d4,i) ¥
	y = d1;	¥
	d1 += ep[i];	¥
	y = LROTATE(y,13); ¥
	z = d1;	¥
	tmp = y;	¥
	y *= ep[(i)+1];	¥
	z &= 511;	¥
	z = sp[z];	¥
	y = LROTATE(y,5);	¥
	z ^= y;	¥
	d1 = LROTATE(d1,y);¥
	y = LROTATE(y,5);	¥
	d3 += d1;	¥
	z ^= y;	¥
	z = LROTATE(z,y);	¥
	d2 += z;	¥
	d4 ^= y;	¥
	d1 = tmp;

#define InvCoreRound(d1, d2, d3, d4, i) ¥
	y = d1;	¥
	d1 = RROTATE(d1,13);¥
	y *= ep[(i)+1];	¥
	tmp = d1;	¥
	d1 += ep[i];	¥
	z = d1;	¥
	y = LROTATE(y,5);	¥
	z &= 511;	¥
	z = sp[z];	¥
	d1 = LROTATE(d1,y); ¥
	z ^= y;	¥
	y = LROTATE(y,5);	¥
	d3 -= d1;	¥
	z ^= y;	¥
	d1 = tmp;	¥
	z = LROTATE(z,y);	¥
	d4 ^= y;	¥
	d2 -= z;

/* The basic mars encryption: (ep is the expanded key array) */
void mars_encrypt(DWORD *in, DWORD *out, DWORD *ep)
{
	register DWORD a,b,c,d,y,z;
	DWORD tmp;
	DWORD *sp = S;

	a = in[0];
	b = in[1];
	c = in[2];
```

```
        d = in[3];
        IVT_DEBUG(a,b,c,d);

#ifndef NO_MIX
        /* first, add subkeys to all input data words */
        a += ep[0];
        b += ep[1];
        c += ep[2];
        d += ep[3];
        IVT_DEBUG(a,b,c,d);

        /* then do eight mixing rounds */
        MixForwardRound(a,b,c,d);
        a += d;
        IVT_DEBUG(a,b,c,d);
        MixForwardRound(b,c,d,a);
        b += c;
        IVT_DEBUG(a,b,c,d);
        MixForwardRound(c,d,a,b);
        IVT_DEBUG(a,b,c,d);
        MixForwardRound(d,a,b,c);
        IVT_DEBUG(a,b,c,d);

        MixForwardRound(a,b,c,d);
        a += d;
        IVT_DEBUG(a,b,c,d);
        MixForwardRound(b,c,d,a);
        b += c;
        IVT_DEBUG(a,b,c,d);
        MixForwardRound(c,d,a,b);
        IVT_DEBUG(a,b,c,d);
        MixForwardRound(d,a,b,c);
        IVT_DEBUG(a,b,c,d);
#endif

        /* then sixteen mars encrypting rounds                    *
         * (eight in forward- and eight in backwards-mode) */

        CoreRound(a,b,c,d,4);
        IVT_DEBUG(a,b,c,d);
        CoreRound(b,c,d,a,6);
        IVT_DEBUG(a,b,c,d);
        CoreRound(c,d,a,b,8);
        IVT_DEBUG(a,b,c,d);
        CoreRound(d,a,b,c,10);
        IVT_DEBUG(a,b,c,d);

        CoreRound(a,b,c,d,12);
        IVT_DEBUG(a,b,c,d);
        CoreRound(b,c,d,a,14);
        IVT_DEBUG(a,b,c,d);
        CoreRound(c,d,a,b,16);
        IVT_DEBUG(a,b,c,d);
        CoreRound(d,a,b,c,18);
        IVT_DEBUG(a,b,c,d);

        CoreRound(a,d,c,b,20);
        IVT_DEBUG(a,b,c,d);
        CoreRound(b,a,d,c,22);
```

```
        IVT_DEBUG(a,b,c,d);
        CoreRound(c,b,a,d,24);
        IVT_DEBUG(a,b,c,d);
        CoreRound(d,c,b,a,26);
        IVT_DEBUG(a,b,c,d);

        CoreRound(a,d,c,b,28);
        IVT_DEBUG(a,b,c,d);
        CoreRound(b,a,d,c,30);
        IVT_DEBUG(a,b,c,d);
        CoreRound(c,b,a,d,32);
        IVT_DEBUG(a,b,c,d);
        CoreRound(d,c,b,a,34);
        IVT_DEBUG(a,b,c,d);

#ifndef NO_MIX
        /* then do eight inverse-mixing rounds */
        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(b,c,d,a);
        c -= b;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(c,d,a,b);
        d -= a;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(a,b,c,d);

        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(b,c,d,a);
        c -= b;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(c,d,a,b);
        d -= a;
        IVT_DEBUG(a,b,c,d);
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(a,b,c,d);

        /* subtract final subkeys */
        a -= ep[2*NUM_ROUNDS+4];
        b -= ep[2*NUM_ROUNDS+5];
        c -= ep[2*NUM_ROUNDS+6];
        d -= ep[2*NUM_ROUNDS+7];
        IVT_DEBUG(a,b,c,d);
#endif

        out[0] = a;
        out[1] = b;
        out[2] = c;
        out[3] = d;
}

/* mars decryption is simply encryption in reverse */
void mars_decrypt(DWORD *in, DWORD *out, DWORD *ep)
{
        register DWORD a,b,c,d,y,z;
        DWORD tmp;
        DWORD *sp = S;
```

```
        d = in[0];
        c = in[1];
        b = in[2];
        a = in[3];
        IVT_DEBUG(d,c,b,a);

#ifndef NO_MIX
        /* first, add subkeys to all input data DWORDs */
        a += ep[2*NUM_ROUNDS+7];
        b += ep[2*NUM_ROUNDS+6];
        c += ep[2*NUM_ROUNDS+5];
        d += ep[2*NUM_ROUNDS+4];
        IVT_DEBUG(d,c,b,a);

        /* then do eight mixing rounds */
        MixForwardRound(a,b,c,d);
        IVT_DEBUG(d,c,b,a);
        a += d;
        MixForwardRound(b,c,d,a);
        IVT_DEBUG(d,c,b,a);
        b += c;
        MixForwardRound(c,d,a,b);
        IVT_DEBUG(d,c,b,a);
        MixForwardRound(d,a,b,c);
        IVT_DEBUG(d,c,b,a);

        MixForwardRound(a,b,c,d);
        IVT_DEBUG(d,c,b,a);
        a += d;
        MixForwardRound(b,c,d,a);
        IVT_DEBUG(d,c,b,a);
        b += c;
        MixForwardRound(c,d,a,b);
        IVT_DEBUG(d,c,b,a);
        MixForwardRound(d,a,b,c);
        IVT_DEBUG(d,c,b,a);
#endif

        /* then sixteen mars decrypting rounds                  *
         * (eight in forward- and eight in backwards-mode) */

        InvCoreRound(a,b,c,d,34);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(b,c,d,a,32);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(c,d,a,b,30);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(d,a,b,c,28);
        IVT_DEBUG(d,c,b,a);

        InvCoreRound(a,b,c,d,26);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(b,c,d,a,24);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(c,d,a,b,22);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(d,a,b,c,20);
        IVT_DEBUG(d,c,b,a);
```

```
        InvCoreRound(a,d,c,b,18);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(b,a,d,c,16);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(c,b,a,d,14);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(d,c,b,a,12);
        IVT_DEBUG(d,c,b,a);

        InvCoreRound(a,d,c,b,10);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(b,a,d,c,8);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(c,b,a,d,6);
        IVT_DEBUG(d,c,b,a);
        InvCoreRound(d,c,b,a,4);
        IVT_DEBUG(d,c,b,a);

#ifndef NO_MIX
        /* then do eight inverse-mixing rounds */
        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(d,c,b,a);
        MixBackwardsRound(b,c,d,a);
        IVT_DEBUG(d,c,b,a);
        c -= b;
        MixBackwardsRound(c,d,a,b);
        IVT_DEBUG(d,c,b,a);
        d -= a;
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(d,c,b,a);

        MixBackwardsRound(a,b,c,d);
        IVT_DEBUG(d,c,b,a);
        MixBackwardsRound(b,c,d,a);
        IVT_DEBUG(d,c,b,a);
        c -= b;
        MixBackwardsRound(c,d,a,b);
        IVT_DEBUG(d,c,b,a);
        d -= a;
        MixBackwardsRound(d,a,b,c);
        IVT_DEBUG(d,c,b,a);

        /* subtract final subkeys */
        a -= ep[3];
        b -= ep[2];
        c -= ep[1];
        d -= ep[0];
        IVT_DEBUG(d,c,b,a);
#endif

        out[0] = d;
        out[1] = c;
        out[2] = b;
        out[3] = a;
}

/*****************************************************************************
 *
```

```
 *     NIST High Level key setup, block encrypt and decrypt routines
 *
 ************************************************************************/

/* table for rapid, case insensitive hex conversion */
BYTE hex[256] = {
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 0, 0, 0, 0, 0,
      0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

/* NIST defined high level key setup */
int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial)
{
      DWORD tmpkey[EKEY_DWORDS];
      int i,j;

      /* sanity check pointers */
      if (key == NULL)
            return(BAD_KEY_INSTANCE);
      if (keyMaterial == NULL)
            return(BAD_KEY_MAT);

      /* save parameters into keyInstance */
      key->direction = direction;
      key->keyLen = keyLen;
      for (i=0;i<keyLen/4;i++)
            key->keyMaterial[i] = keyMaterial[i];
      key->keyMaterial[MAX_KEY_SIZE] = '¥0';

      /* convert ascii keyMaterial to BYTEs */
      for(i=0,j=0;i<keyLen/4;i+=2,j++)
            ((BYTE *)tmpkey)[j] = (BYTE)((hex[(int)keyMaterial[i]]<<4) |
                                         hex[(int)keyMaterial[i+1]]);
#     ifdef SWAP_BYTES
            /* BSWAP the input key DWORDs */
            for (i=0;i<keyLen/W;i++)
                  tmpkey[i] = BSWAP(tmpkey[i]);
#     endif

      /* call low level mars setup routine */
      return(mars_setup(keyLen/32,tmpkey,key->E));
}

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV)
{
      int i,j;
```

```
    /* sanity check pointers */
    if (cipher == NULL)
        return(BAD_CIPHER_MODE);

    /* save cipher parameters */
    cipher->mode = mode;

    /* handle IV */
    if((mode == MODE_CBC)||(mode == MODE_CFB1)) {
        if(IV == NULL)
            return(BAD_CIPHER_MODE);
        /* convert ascii IV to BYTEs */
        for(i=0,j=0;j<MAX_IV_SIZE;i+=2,j++)
            cipher->IV[j] = (BYTE)((hex[(int)IV[i]]<<4) | hex[(int)IV[i+1]]);
        /* copy BYTE IV to DWORD CIV, with conversion if necessary */
        for(i=0;i<NUM_DATA;i++)
            cipher->CIV[i] = BSWAP(((DWORD *)&cipher->IV)[i]);
    }
    return(TRUE);
}

/* this assumes the input length is a multiple of 128 bits */
int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer)
{
    DWORD tmp[4];
    int i;

    if (cipher->mode == MODE_ECB) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#           else
                mars_encrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
#           endif
        }
    }
    else if(cipher->mode == MODE_CBC) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                tmp[0] = BSWAP(*(DWORD *)(input+i+0)) ^ cipher->CIV[0];
                tmp[1] = BSWAP(*(DWORD *)(input+i+4)) ^ cipher->CIV[1];
                tmp[2] = BSWAP(*(DWORD *)(input+i+8)) ^ cipher->CIV[2];
                tmp[3] = BSWAP(*(DWORD *)(input+i+12)) ^ cipher->CIV[3];
                mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                cipher->CIV[0] = *(DWORD *)(outBuffer+i+0);
                cipher->CIV[1] = *(DWORD *)(outBuffer+i+4);
                cipher->CIV[2] = *(DWORD *)(outBuffer+i+8);
                cipher->CIV[3] = *(DWORD *)(outBuffer+i+12);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
```

```
                    *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                    *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                    *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#               else
                    tmp[0] = *(DWORD *)(input+i+0) ^ cipher->CIV[0];
                    tmp[1] = *(DWORD *)(input+i+4) ^ cipher->CIV[1];
                    tmp[2] = *(DWORD *)(input+i+8) ^ cipher->CIV[2];
                    tmp[3] = *(DWORD *)(input+i+12) ^ cipher->CIV[3];
                    mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                    cipher->CIV[0] = *(DWORD *)(outBuffer+i+0);
                    cipher->CIV[1] = *(DWORD *)(outBuffer+i+4);
                    cipher->CIV[2] = *(DWORD *)(outBuffer+i+8);
                    cipher->CIV[3] = *(DWORD *)(outBuffer+i+12);
#               endif
            }
        }
        else if(cipher->mode == MODE_CFB1) {
                    cipher->mode = MODE_ECB;        /* do encryption in ECB */
                    for (int n=0;n<inputLen;n++)
                            {
                            blockEncrypt(cipher,key,(BYTE      *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                            bit0    = 0x80 >> (n & 7);/* which bit position in byte */
                            ctBit = (input[n/8] & bit0) ^ ((((BYTE *) x)[0] & 0x80) >> (n&7));
                            outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) | ctBit;
                            carry = ctBit >> (7 - (n&7));
                            for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                    {
                                    bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                    cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                    carry = bit;
                                    }
                            }
                    cipher->mode = MODE_CFB1;       /* restore mode for next time */
                    return inputLen;
            /* Input is one bit (lsb of first byte).
              * Encrypt IV with key, xor input with msb of encrypted IV,
              * and then feed output cipher bit into lsb of IV.
              */
                    /*
            DWORD ECIV[4];

            if(inputLen != 1)
                return(BAD_CIPHER_MODE);

            mars_encrypt(cipher->CIV, ECIV, key->E);
            outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
            cipher->CIV[0] = (cipher->CIV[0]<<1) | (cipher->CIV[1] & 0x80000000);
            cipher->CIV[1] = (cipher->CIV[1]<<1) | (cipher->CIV[2] & 0x80000000);
            cipher->CIV[2] = (cipher->CIV[2]<<1) | (cipher->CIV[3] & 0x80000000);
            cipher->CIV[3] = (cipher->CIV[3]<<1) | (DWORD)outBuffer[0];*/
        }
        else
            return(BAD_CIPHER_MODE);

        return(inputLen);
}

/* this assumes the input length is a multiple of 128 bits */
```

```
int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer)
{
    int i;

    if (cipher->mode == MODE_ECB) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                DWORD tmp[4];
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_decrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#           else
                mars_decrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
#           endif
        }
    }
    else if(cipher->mode == MODE_CBC) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                DWORD tmp[4];
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_decrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0)
                    ^ cipher->CIV[0]);
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4)
                    ^ cipher->CIV[1]);
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8)
                    ^ cipher->CIV[2]);
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12)
                    ^ cipher->CIV[3]);
                cipher->CIV[0] = tmp[0];
                cipher->CIV[1] = tmp[1];
                cipher->CIV[2] = tmp[2];
                cipher->CIV[3] = tmp[3];
#           else
                mars_decrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) ^= cipher->CIV[0];
                *(DWORD *)(outBuffer+i+4) ^= cipher->CIV[1];
                *(DWORD *)(outBuffer+i+8) ^= cipher->CIV[2];
                *(DWORD *)(outBuffer+i+12) ^= cipher->CIV[3];
                cipher->CIV[0] = *(DWORD *)(input+i+0);
                cipher->CIV[1] = *(DWORD *)(input+i+4);
                cipher->CIV[2] = *(DWORD *)(input+i+8);
                cipher->CIV[3] = *(DWORD *)(input+i+12);
#           endif
        }
    }
    else if(cipher->mode == MODE_CFB1){
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
```

```
                    for (n=0;n<inputLen;n++)
                            {
                            blockEncrypt(cipher,key,(BYTE       *)cipher->IV,BLOCK_SIZE,(BYTE
*)x);
                            bit0    = 0x80 >> (n & 7);
                            ctBit = input[n/8] & bit0;
                            outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) |
                                                    (ctBit ^ ((((BYTE *) x)[0] & 0x80)
>> (n&7)));
                            carry = ctBit >> (7 - (n&7));
                            for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                    {
                                    bit = cipher->IV[i] >> 7;    /* save next "carry" from shift */
                                    cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                    carry = bit;
                                    }
                            }
                    cipher->mode = MODE_CFB1;       /* restore mode for next time */
                    return inputLen;


            /* Input is one bit (lsb of first byte).
              * Encrypt IV with key, xor input with msb of encrypted IV,
              * and then feed input cipher bit into lsb of IV.
              */
                    /*
            DWORD ECIV[4];

            if(inputLen != 1)
                return(BAD_CIPHER_MODE);

            mars_encrypt(cipher->CIV, ECIV, key->E);
            outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
            cipher->CIV[0] = (cipher->CIV[0]<<1) | (cipher->CIV[1] & 0x80000000);
            cipher->CIV[1] = (cipher->CIV[1]<<1) | (cipher->CIV[2] & 0x80000000);
            cipher->CIV[2] = (cipher->CIV[2]<<1) | (cipher->CIV[3] & 0x80000000);
            cipher->CIV[3] = (cipher->CIV[3]<<1) | (DWORD)(input[0]&1);
*/
        }
        else
            return(BAD_CIPHER_MODE);

        return(inputLen);
}


//////////////////////////////////////////////////////////

// MarsDC.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

#include "stdafx.h"

#include <stdio.h>
#include <time.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <malloc.h>
```

```
#include "aes.h"

#define file_len(x) (unsigned long)x

int s;
keyInstance keyI;
cipherInstance cipherI;

int main(int argc, char* argv[])
{
	FILE *fkey , *fin , *fout ;
    fpos_t flen;
    unsigned long len, rlen, blen4, pfilelen;
	int mode,klen,blen,rc;
	char c_mode[4], c_klen[8], c_key[128], c_cini[64];
	unsigned char cbuf[2048], pbuf[2048];

	blen4 = 2048;

	printf( "Start!¥n" );

	if( argc != 4 ){                                    // 使い方の誤り
		printf( "argc != 4 ¥n" );
		return -1;
	}

	if((fkey = fopen(argv[1], "rt")) == NULL){
		printf("Can not find key file for decryption. ¥n");
		return (-1);
	}

	fgets( c_mode, 4, fkey );
	fgets( c_klen, 8, fkey );
	fgets( c_key, 128, fkey );
	fgets( c_cini, 64, fkey );

	mode = atoi(c_mode);
	klen = atoi(c_klen);
	blen = 128;

	if(klen<56 || 448<klen){
		printf("Wrong key size. ¥n");
		return (-1);
	}

	/*Set mode*/
	if(mode == 1){
		rc=cipherInit(&cipherI, MODE_ECB, "");
	}
	if(mode == 2){
		rc=cipherInit(&cipherI, MODE_CBC, c_cini);
	}
	if(mode == 3){
		rc=cipherInit(&cipherI, MODE_CFB1, c_cini);
	}
	if(rc<=0){
		printf("モード設定が出来ません。");
		return(-2);
	}
```

```
        makeKey(&keyI, DIR_DECRYPT,    klen, c_key );


        if((fin = fopen(argv[2], "rb")) == NULL){
                printf("Can not open crypted file. ¥n");
                return (-1);
        }
        if((fout = fopen(argv[3], "wb")) == NULL){
                printf("Can not open plane data file. ¥n");
                return (-1);
        }

    fseek(fin, 0, SEEK_END);
    fgetpos(fin, &flen);
    rlen = file_len(flen);
    // reset to start
    fseek(fin, 0, SEEK_SET);
        s = sizeof(unsigned long);
        // write the bytes of the file
        len = (unsigned long) fread(cbuf, 1, blen4 , fin);
        rlen = rlen - len;

        if(len < blen4){ return -3;  }

        rc=blockDecrypt(&cipherI,          &keyI,(unsigned          char*)(cbuf),8*blen4,(unsigned
char*)(pbuf));
                // 復号文出力
        pfilelen = *((long*)(pbuf));

        if(pfilelen <= blen4 - s){
                if(fwrite(pbuf+s, 1, pfilelen, fout) != pfilelen){return -2;}
                if(fkey){ fclose(fkey);         }
                if(fout){ fclose(fout);         }
                if(fin)    { fclose(fin);       }
                printf( "End!¥n" );
                return 0;
        }
        else{
                if(fwrite(pbuf+s, 1, blen4 - s, fout) != blen4 - s){return -2;}
                pfilelen -= (blen4 - s);
        }

    if((rlen <= blen4) && (rlen > 0))
        {           // if the file length is less than or equal to 2048 bytes
        len = (unsigned long) fread(cbuf, 1, blen4 , fin);
        rlen -= len;
        if(rlen > 0){ return -2; }
                rc=blockDecrypt(&cipherI,    &keyI,(unsigned    char*)(cbuf),8*blen4,(unsigned
char*)(pbuf));
                if(fwrite(pbuf, 1, pfilelen, fout) != pfilelen){ return -2; }
                if(fkey){ fclose(fkey);         }
                if(fout){ fclose(fout);         }
                if(fin)    { fclose(fin);       }
                printf( "End!¥n" );
                return 0;
    }
    else
    {    // if the file length is more 1024 bytes
```

```
        // read the file a block at a time
        while(rlen > 0 && !feof(fin))
        {
            // read a block and reduce the remaining byte count
            len = (unsigned long)fread(cbuf, 1, blen4 , fin);
            rlen -= len;
            if((rlen>0) && (len==blen4)){
                                    rc=blockDecrypt(&cipherI,                   &keyI,(unsigned
char*)(cbuf),8*blen4,(unsigned char*)(pbuf));
                                    if(fwrite(pbuf, 1, blen4, fout) != blen4){return -2;}
                                    pfilelen -= blen4;
                            }
            if(rlen<=0){
                                    rc=blockDecrypt(&cipherI,                   &keyI,(unsigned
char*)(cbuf),8*blen4,(unsigned char*)(pbuf));
                                    if(fwrite(pbuf, 1, pfilelen, fout) != pfilelen){return -2;}
                                    if(fkey){ fclose(fkey);     }
                                    if(fout){ fclose(fout);     }
                                    if(fin)   { fclose(fin);    }
                                    printf( "End!¥n" );
                                    return 0;
                            }
        }
    }
        if(fkey){ fclose(fkey);     }
        if(fout){ fclose(fout);     }
        if(fin)   { fclose(fin);    }
        printf( "End!¥n" );
        return 0;
}


//////////////////////////////////////////////////

// stdafx.cpp：標準インクルードファイルを含むソース ファイル
//                  MarsDC.pch 生成されるプリコンパイル済ヘッダー
//                  stdafx.obj 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"

// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。

//////////////////////////////////////////////////


/*  aes.h  */

/*  AES Cipher header file for ANSI C Submissions
 *      Lawrence E. Bassham III
 *      Computer Security Division
 *      National Institute of Standards and Technology
 *
 *      April 15, 1998
 *
 *      Modified for IBM submission
 *      David Safford
 *      4/16/1998
 */
```

```
#include <stdio.h>

/*****************************************************************************
 *
 *      NIST High Level C API, with some IBM additions
 *
 *****************************************************************************/

#define TRUE                1
#define FALSE               0

#define DIR_ENCRYPT         0     /*  Are we encrypting?                          */
#define DIR_DECRYPT         1     /*  Are we decrypting?                          */
#define MODE_ECB            1     /*  Are we ciphering in ECB mode?               */
#define MODE_CBC            2     /*  Are we ciphering in CBC mode?               */
#define MODE_CFB1           3     /*  Are we ciphering in 1-bit CFB mode?         */

#define BAD_KEY_DIR         -1    /*  Key direction is invalid                    */
#define BAD_KEY_MAT         -2    /*  Key material not of correct length          */
#define BAD_KEY_INSTANCE -3   /*  Key passed is not valid                     */
#define BAD_CIPHER_MODE   -4   /*  Params struct passed to cipherInit invalid */
#define BAD_CIPHER_STATE -5   /*  Cipher in wrong state                       */

typedef unsigned char BYTE;

/* IBM Addition - a DWORD must be 32 bits for this implementation */
typedef unsigned long DWORD;

/* IBM specific defines: these parameters can be changed                    */
#define NUM_MIX 8               /* number of mixing rounds per stage        */
#define NUM_ROUNDS 16           /* number of full core rounds               */
#define NUM_SETUP 7             /* number of key setup mixing rounds        */

/* IBM specific defines: these parameters are fixed for this implementation */
#define W               32          /* number of bits in a word             */
#define NUM_DATA        4           /* data block size in words             */
#define EKEY_DWORDS (2*(NUM_DATA+NUM_ROUNDS))       /* number of subkey words
*/

/* IBM modified values                                                      */
#define MAX_KEY_SIZE (EKEY_DWORDS*8)   /* max ASCII char's needed for a key     */
#define MAX_IV_SIZE (NUM_DATA*4)        /* max bytes's needed for an IV         */

/*  The structure for key information */
typedef struct {
    BYTE    direction;          /*  Key used for encrypting or decrypting?  */
    int     keyLen;             /*  Length of the key in BITS               */
    char    keyMaterial[MAX_KEY_SIZE+1];   /*  Raw key data in ASCII           */
    DWORD   E[EKEY_DWORDS];     /* IBM addition for mars expanded key       */
} keyInstance;

/*  The structure for cipher information */
typedef struct {
    BYTE mode;                  /*  MODE_ECB, MODE_CBC, or MODE_CFB1        */
    BYTE IV[MAX_IV_SIZE];   /*  initial binary IV BYTE for chaining     */
    DWORD CIV[NUM_DATA];        /*  IBM addition: current IV in binary WORDs */
} cipherInstance;
```

```
/*    NIST High level function protoypes    */
int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial);

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                        int inputLen, BYTE *outBuffer);

int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                        int inputLen, BYTE *outBuffer);

/*****************************************************************************
 *
 *       IBM Low Level (WORD Oriented) API
 *
 *****************************************************************************/

/* setup a mars expanded key
 *
 * k    (input) is the number of words in the key
 * kp (input) is a pointer to the array of k key words
 * ep (output) is a pointer to an array of EKEY_WORDS expanded subkey WORDs
 */
int mars_setup(int k, DWORD *kp, DWORD *ep);

/* The basic mars encryption of one block (of NUM_DATA DWORDS) */
void mars_encrypt(DWORD *in, DWORD *out, DWORD *ep);

/* mars decryption is simply encryption in reverse */
void mars_decrypt(DWORD *in, DWORD *out, DWORD *ep);

////////////////////////////////////////////////////////////

// stdafx.h : 標準のシステム インクルード ファイル、
//                  または参照回数が多く、かつあまり変更されない
//                  プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__F43EC674_C5E9_41BE_88F7_6EC08666AC44__INCLUDED_)
#define AFX_STDAFX_H__F43EC674_C5E9_41BE_88F7_6EC08666AC44__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif
// !defined(AFX_STDAFX_H__F43EC674_C5E9_41BE_88F7_6EC08666AC44__INCLUDED_)
```

１０．５．３　鍵の形式

3
448
02FFAD21FDB3E0DC0DF9EA6C97F470B18437CE6638E2AE5DC80EDA0C676C4E4922289252CE1446103E02FE9DB9715103AA6D2D479AB06DA6
0123456789abcdef0123456789abcdef

のような形になります。上の例では、CFB1 モードで鍵の長さは 448 ビットの場合の例です。

これらの暗号ソフトを、簡単に 5 段階まで適用して 5 重に暗号化できます。この Protected Communication System（Mail-J）は、米国、欧州の特許を取得した内容をソフトウェアとして実現するものです。（欧州特許は 2007/2/14）
特徴は、
１．送信者と受信者の順序対ごとに暗号化方式を決定できる。
２．様々な暗号技術を多重に適用して通信（メール）の秘密を守ります。
３．鍵は利用者が自由に作成する
４．最新の暗号技術の発展に合わせて使用する暗号化ソフトを簡単に変更できます。
となります。
このソフトによって、アドレス帳に暗号化ソフト名、暗号化鍵のファイル名、複号化ソフト名、復号化鍵のファイル名を登録するだけで、様々な暗号技術を簡単に利用できるようになります。
現在は、７種類（RSA, AES, Camellia, Twofish, Serpent, MARS, 他）が利用可能ですが、楕円曲線暗号を追加する予定です。
また、暗号ソフトをどのような形式で作成したらこのソフトで利用可能となるかは上の例を見れば理解できると思います従って、独自に開発した暗号を利用するのも簡単にできます。
暗号ソフトとメーラーは完全に別のソフトです。したがって、暗号を開発する人に対して適正な利益が得られる基盤が提供されます。この結果、沢山の暗号方式が発表され、個人的な情報を簡単に、しかも強力に暗号化して送信できるようになります。利用者は合理的な出費で、暗号強度が十分であるものを利用できるようになります。
後ほど、暗号ソフトをどのように作成したらこのソフトで利用できて、販売が可能となるのかを詳しく示しますので、多くの暗号開発者の方々が参加されますようにお願いいたします。
DES や Triple-DES の安全性が疑問視されて、新しい暗号 AES が募集されました。RSA での鍵の長さは 1024 ビット以上が必要とされています。このように、現在では暗号技術の変化がとても速いので、最新の理論や技術を利用した暗号化の方式も明日には解読されているかもしれません。この問題に対して３つの方法で対応します。
１．最新の技術や理論を使った暗号化ソフトを簡単に組み込める仕組みを最初から用意します。
２．多重暗号化によって解読を困難にします。
３．鍵は利用者が自由に作成する。
この 3 つの方法によって、あなたの個人情報を盗聴から守ります。

１１．　暗号ソフトを作成される方に

１１．１　使用できる暗号の種類と特徴：

暗号には、大きく分けて秘密鍵方式と公開鍵方式の２種類がありますが、どちらも利用可能です。
理論上は、暗号化鍵と暗号化ソフトを分離しなくても良いのですが、ハードウエアの現状に適しているとの理由で、暗号化鍵と暗号化ソフトを分離する方式について記述します。
暗号用の商用ソフトは、鍵生成ソフトと、暗号化ソフト、復号化ソフトのセットで、次のような構造を持つものとして制作されるべきです。
（１）暗号化鍵の作成機能をもち、暗号化鍵の自由な配布が認められている。
（２）暗号化ソフトの自由な再配布が認められている。

（3）復号化鍵の作成機能をもつ。
（4）復号化ソフトの再配布を禁止できる。
この、1から4を満たすようになっていれば秘密鍵方式でも、公開鍵方式でもどちらも利用することができます。このような構造で暗号ソフトを制作することは困難ではありません。
復号化ソフトの再配布が禁止できれば、暗号ソフトの利用者がそれぞれにソフトを購入することになりますので、商業的にソフトを制作販売することが可能となります。より優れた暗号ソフトの開発が可能となるのです。
皆さま一人一人が暗号ソフトを自作しても良いのですが、安全な暗号ソフトを作成するのはかなりの時間と努力を必要とします。暗号ソフトは既にかなりの種類が存在し、専門家も作った社会的評価の高いものを選ぶのが現実的な選択です。
暗号化と復号化のソフトに関する要求。
1．名前の変更が可能であること。
2．引数を3つもっていること。
3．暗号化ソフトの場合は、第1引数は、暗号化で使用する鍵を記録したファイル名、第2引数はこれから暗号化されるファイル名、第3引数は暗号化された結果のファイル名。
4．復号化の場合は、第1引数は、復号化で使用する鍵を記録したファイル名、、第2引数は（暗号化されている）これから復号化するファイル名、第3引数は復号化されたデータを記録するファイル名
暗号化ソフト、復号化ソフトは、コンソールタイプのアプリケーションとして開発して下さい。鍵生成ソフトは GUI タイプにすると良いと思います。

暗号化、復号化のときの動きは次のようにして下さい。

暗号化の場合には、
暗号化ソフト(encrypted.exe)、暗号化鍵(ekey.bin)、平文ファイル(plane.bin)、暗号化したファイル(encrypted.bin)とするとき、MS-DOS ボックスから
encrypt(ekey.bin, plane.bin, encrypted.bin)
と呼び出したときには、2つのファイル(ekey.bin、plane.bin)をバイナリファイルとして読み込んで、与えられたファイル名(encrypted.bin)の暗号化されたバイナリファイルを生成する様に機能しなくてななりません。
復号化の場合には、
復号化ソフト(decrypt.exe)、暗号化鍵(dkey.bin)、暗号化したファイル(encrypted.bin)、平文ファイル(plane.bin)、とするとき、MS-DOS ボックスから
decrypt(dkey.bin, encrypted.bin, plane.bin)
と呼び出したときには、2つのファイル(dkey.bin, encrypted.bin)をバイナリファイルとして読み込んで、与えられたファイル名(plane.bin)の復号化されたバイナリファイルを生成する様に機能しなくてなりません。
暗号化や復号化が終了したら自動的に終了しなくてなりません。コンソールタイプならそのように機能します。また計算過程を表示させるようにした方が、利用者が楽しめると思います。16 進数で表示すると良いでしょう。暗号ソフトが多重暗号化のどの場面で利用されるかは分かりませんので、テキスト表示では問題が生じます。

暗号化、復号化のソフトを作成される場合には、データの長さにご注意ください。ブロック型の暗号ではブロック長の整数倍となるようにデータを加えて暗号化するので、どこかに元のデータサイズを記載し、復号時にはその数値を使って、元のデータサイズに調整する必要があります。
また、データファイルを開くときはバイナリデータとして開いてください。理由は、多重暗号化の場合、一度暗号化されたものはバイナリーデータとして扱うことが必要だからです。データはバイナリーデータとして開いて、バイナリーデータとして保存してください。
鍵生成ソフトは、共通鍵方式の時も名前を変えた2つのかぎファイルを生成するようにして下さい。ダイアログボックスから暗号化鍵と復号化鍵のファイル名をそれぞれ入力できる様にして下さい。
鍵が単なる乱数のような場合は、暗号ソフトの利用者が自分で乱数を生成すればそれで済みますが、RSA の様な場合には、単純な乱数の組では機能しません。鍵生成ソフトが必要になります。鍵生成ソフトが認証用のファイルがないと動かないようにしておいて、認証用ファイルの入手時にソフト代を支払う様にすれば暗号ソフトの制作をする人にも収入が得られる用になります。もちろん、本来は単純な乱数で動く暗号化ソフトでも、鍵ファイルの中に埋め込まれた特別な値を確認してから動くように作っておけば、その特別な値を持っている鍵ファイルを作成する為の鍵生成ソフトが必要になります。この鍵生成ソフトを起動させるときに認証用ファイルが必要であるなら、その認証用ファイルを入手するときに、代金を支払ってもらうことができます。暗号ソフトを制作し販売される方々が、ユーザーが利用しやすい価格で販売して下さるようにお願いいたし

ます。

　暗号メーラーの構造に関しては､アメリカ､ヨーロッパで特許取得していますので､暗号メーラーの作成が、私の特許権に抵触する可能性があります。ご注意ください。

１１．２　暗号ソフトの呼び出しコード

暗号ソフトは次の様に呼び出されますので、参考にしてください。

1. Encryption（暗号化）では、
```
if((strlen(ecp1)>0)&&(strlen(eck1)>0)){
ret = _spawnl(_P_WAIT, ecp1, ecp1, eck1, ecpath0, ecpath1, NULL);
}else{
CopyFile(ecpath0,ecpath1, FALSE);
}
```
の様に呼び出します。

2. Decryption（復号化）では、
```
if((strlen(dcp5)>0)&&(strlen(dck5)>0)){
ret = _spawnl(_P_WAIT, dcp5, dcp5, dck5, dcpath5, dcpath4, NULL);
}else{
CopyFile(dcpath5, dcpath4, FALSE);
}
```
の様に呼び出します。

１２．価格

　Protected communication system(MailJ)の販売はソフトハウス・ナオがおこなっていますので、購入方法や価格については
SoftHouse_Nao@yahoo.co.jp
まで、メールでお問い合わせ下さい。

１３．利用しているフリーソフトについて

　文字コードの変換には
　株式会社ピーデーの　川俣　晶　氏が作成された DLL を利用しています。
この DLL の 配布規則は

　本ソフトウェアは、株式会社ピーデーがサービスのために配布するもので、商品ではありません。無料です。従って、一切のサポートはありません。保証もありません。

　なんらかの保証が必要とされる場合は、内容に関する有償サポート契約が可能です。詳しくは、support@piedey.co.jp までご連絡下さい。

　本ソフトウェアの改編、転載などは、商用、非商用に関わらず自由です。ただし、文字コード変換機能に関する著作権表記"Copyright (c) 1999,2000 by Pie Dey Co.,Ltd."を行うことを条件とします。

　上記の規則より、tconvlib.dll を他のアプリケーションソフトウェアの一部として再配布することが可能ですが、その場合は、バージョン間の非互換性の問題を回避するために、必ずアプリケーションソフトウェア固有のディレクトリにインストールし、System ディレクトリなどの共用ディレクトリに保存しないようにして下さい。

　本ソフトウェアの配布アーカイブをそのままパソコン通信、インターネット、

書籍などに掲載する場合は、ご一報下さい。tconvlib.dll をソフトウェアの一
部として再配布するような場合には、連絡は不要です。

にしたがって、利用させていただいています。ソフトウエアの一部として tconvlib.dll を使わせていただいています。

この DLL はこのメーラーに不可欠のものでした。上記のような形で提供してくださったことに関して深く感謝いたします。ありがとうございます。

なお、著作権表示はバージョン情報の一部として記載しています。

*

１４．Protected Communication System (MailJCM)の作成

１４．１　開発方針

私的なメールの内容は重要な個人情報であると考えます。従って個人情報保護の精神を実現した通信を電子メールと電話の両方で実現しようと考えました。この考えに具体性を持たせて、特許申請しました。この特許の内容を実現すれば世界最強の暗号通信システムが実現できると信じてこれを作成しました。

盗聴に対抗できる安全な通信方法を実現する方法として、Protected Communication System を考え始めたのは 2000 年 4 月でした。これを特許申請して、2007 年 2 月 14 日にヨーロッパ特許庁からの正式認定を受け取りました。また、アメリカにおける特許も認められました。電子メールと電話に関しての特許認定を受けています。すべてのメールとすべての電話（ＩＰ電話）による通信を意味のある形で暗号化することを目標に開発をしています。

この通信方法をメーラーとして実現したものがこれです。したがって、このメーラーの目的は、---“すべてのメールでの通信を意味のある形で暗号化する。”---ことです。

意味のある暗号化とは何を意味するのでしょうか？それには意味のない暗号化と比べればすぐに答えが出ます。結論は

１．暗号化鍵を他人に預けない。
２．鍵の長さに制限を受けない。
３．アルゴリズムの種類を限定されず、最新アルゴリズムがすぐに利用できる。
４．暗号化、復号化はサーバーではなく自分の端末機と相手の端末機で行ない、平文データが流れる経路を無くす。
５．十分な多重暗号化が可能である。
６．暗号アルゴリズムは送信者と受信者の順序対ごとに独立している。

となります。

アメリカでは、2002 年に標準暗号化の方式を Triple-DES から、AES に変更しました。変更の理由は解読されてしまうことと、暗号を利用する場面が広がったからです。

アメリカでは、新しい暗号を求めて世界中の研究者に呼びかけて強力でしかも、ライセンス料を払わないで使用できる暗号を募集しました。この結果 AES として使用される暗号が Rijndael に決まりました。

AES の特徴は暗号化するときのブロック長が 128 ビット以上であることです。世界中からの応募の中で、Rijndael が最終的に選ばれました。この選考の最終段階まで残っていたものに、Serpent, Twofish, MARS があります。解読可能なものは意味がありません。解読できないもの、新しい技術を採用したものを利用しなくてはなりません。幸いにして、これらは無料で使用できます。４つとも CD に入っています。さらに、日本の三菱電気、NTT が共同で開発した Camellia 暗号も入っています。

AES は対称鍵方式ですので、お互いに鍵を交換するには、郵送や直接あって交換するか、または公開鍵暗号方式で、お互いの対称鍵を交換しなくてはなりません。

幸いなことに、RSA の特許が切れました。これを利用すれば良さそうなものですが、次に問題となるのは暗号化鍵の長さです。RSA は 512 ビットあれば安心といわれていたのはつい最近です。今は最低でも 1024 ビットは必要だと言われています。長い鍵を自由に作れなくてはなりません。1024 ビットで安心できるのは 2010 年までであるとも言われています。

鍵の長さは 2 つの制限を受けます。1 つは法律上の制限。もう一つはコンピュータの能力による制限です。インターネットに掲載する場合や、輸出する場合では、512 ビット以内という強い制限がつきます。これは実

際上使い物にはなりません。幸いにも日本国内での配布と利用には法律的な制限はありません。したがって長い鍵を利用できる RSA 暗号は日本国内でのみ、配布および利用できることになります。

CD には、長い鍵が作成できて、しかも公開鍵の部分を暗号化ソフトと共に、日本国内において自由に配布できるものが入っています。しかしながら、長い鍵を使うと計算時間がとても長くかかります。普通のパソコンの能力ではストレスがたまってしまうでしょう。

最新のアルゴリズムは楕円曲線暗号です。これは、鍵の長さが短くても強力な暗号化が可能になります。残念ながら、現在は開発中です。

暗号開発と暗号解読はいっしょに発展しています。たとえば、楕円曲線暗号の開発は新しい因数分解アルゴリズムを提供し、RSA を危機に陥れていますが、それは同時に新しい強力な暗号方式、楕円曲線暗号を与えてくれています。

このような中で、安全な通信を実現する方法として、次の機能を持つ通信ソフトが必要であると考えます。

１．最新の暗号アルゴリズムをすぐに利用できる。
２．異なるアルゴリズムでの多重暗号化が可能である。
３．暗号化、復号化の鍵は利用者が自由に作成できる。
４．暗号化ソフトと復号化ソフトの 2 つに分かれていて、暗号化ソフトの再配布は自由であるが復号化部分の再配布は禁止されている。
５．使用する暗号アルゴリズムの設定は、送信者と受信者の各順序対ごとに独立した形で可能。

このような機能を持ったメーラーが、このソフトです。他の通信においてもこの機能は必要であると考えます。現在は、メーラーとしてのみ提供されているのですが、すべての通信が上の１，２，３、４の原則の上に実現されるように願っています。

このソフトに付属する暗号ソフト RAS, AES, Twofish, Camellia, Serpent, MARS の他に 1 つ付いています。これらについて簡単に説明します。このソフトに現在付属するのは、公開鍵方式の RSA と対象鍵方式（秘密鍵方式）のものが 6 種類ですが、本来のソースコードが公開されているものに関してはこのマニュアルの後半にそれらを修正して、この暗号メーラーで使用できる形にしたもののソースコードを記載してあります。このメーラーで使用するために変更した部分もありますので、安全性などに関しては本来の公開されているソースコードと比較検討されて、十分納得された上でご利用ください。

せっかく作るのですから、使用する暗号は最新の強力なものでなくてはなりません。共通鍵方式のものについては、AES の選出過程で最後まで候補として残っていたもの（Rijndael, Serpent, Twofish, MARS など）のソースコードが公開されています。暗号の製作者に連絡とり、無料で使用できるものに関して、私なりに変形したものを作成しました。さらに、日本製の暗号 Camellia についてもソースコードが自由に使えますので利用させていただきました。これだけそろっているので、5 段階の多重暗号化と組み合わせれば送信者と受信者が直接鍵を交換できる場合に関しては充分強力な暗号化を与えてくれると考えます。

公開鍵方式については、RSA を用意してあります。もちろん、ECC が有った方が良いことは十分理解しております。しかし、私が数学的に内容を理解するのに、もう少し時間がかかります。

また、それぞれのアルゴリズムについて、暗号化鍵と復号化鍵を作成するソフトを作りました。それぞれの方は自分の持っているコンピュータの性能と送信しての持っているコンピュータの性能を考慮して鍵の長さや暗号アルゴリズムを決定できます。

送信部分の作成に関しては BlatJ を参考にしました。しかしながら添付ファイルの名前が長かったりカタカナだったりした場合の扱いについては自分で最初から書き上げました。

受信部分に関しては、参考文献
１．
２．
を

個人情報保護は、暗号メールから！！　暗号メールであなたの個人情報を守るメーラーが
Protected Communication System（Mail-J）（EU 特許取得、米国特許申請中）
です。　特徴は
１．多重暗号化で守る。５段階の暗号化か出来ます。
２．最新技術で守る。最新の暗号方式をすぐに採用できます。
３．鍵は自分で作成する。どんな長い鍵でも自由に作れます。
です。これを利用して、あなたのプライバシーを自分で守ってください。

これは、多重暗号化による情報の保護を目的としたメーラーの解説です。

（EU での特許取得しま。米国も特許がとれました。）
　インターネットでの販売は暗号機能が強すぎて出来ません。このメーラーでは、暗号化方式を簡単に変更できることや、使用する暗号化鍵の長さが長いことなどの理由です。
　日本国内では暗号メールに関する法的な制限はありませんので、国内での利用に関しては法的な問題はありません。したがって、日本国内でのみご利用可能です。
　このメーラーは、SoftHouse_Nao　が販売しています。
価格などは、SoftHouse_Nao@yahoo.co.jp　宛てにメールでお問い合わせ下さい。
また、技術的な質問などがありましたら、作者に宛てて気軽にメールして下さい。
（作者のメールアドレスは uyama@kc4.so-net.ne.jp です）　　（C）　宇山　靖政
←サンプルソフトです、お試し下さい。

最新マニュアル＞Protected Communication system（Mail-J）

１４．２　参考文献
１４．３　教えてグー
１４．４　バグ取り

１５．IP 電話
１５．１　開発方針

１５．２　参考文献