H11 - セキュリティWG－3

暗号利用技術ハンドブック
第２版

平成 12 年３月

電子商取引実証推進協議会

セキュリティWG

暗号利用技術ハンドブック（第２版）

平成12年3月

電子商取引実証推進協議会 セキュリティWG

目　　次

1 暗号技術の役割

電子商取引の分野においては、決済に限らず不正防止が信頼性の要となる。不正を防止するためには各種セキュリティ機能を駆使したインフラ作りとシステムの構築が不可欠な物であり、暗号技術は、このような情報セキュリティ機能を実現するための、重要な基礎技術である。

情報セキュリティには以下の４つの基本機能がある。

(1) 機密性（Confidentiality）

簡単に言えば、第三者に情報が漏れないようにすることである。すなわち機密性は、情報の利用を正当な権限のある人にだけ制限する機能である。

(2) 完全性（Integrity）

作成された情報は、作成者あるいは発信者の手元を離れて受信者に届くまでの間に、改変されるようなことがあってはならない。完全性は、このような改変を防止する機能である。

(3) 真正性（Authenticity）

真正性は、情報の作成者がまさに作成者であることを保証する機能、または通信相手が本人自身であることを保証する機能である。

(4) 責任追及性（Accountability）

情報が複数の当事者によって処理される場合、何らかの誤りが生じる可能性がある。誤りが生じたときには修正しなければならないが、修正のために過去にさかのぼって全ての処理をチェックすることが出来るように、行為の責任を明示する機能が責任追及性である。

このようなセキュリティ機能を実現するに際して、暗号技術は重要な基礎技術である。具体的には第一番目の「機密性」実現のために、「暗号技術（Cryptography）」と「情報秘匿（Stenography）」の２種類の技術を使うことができる。暗号技術は、さらに共通鍵暗号方式と公開鍵暗号方式の二つに分かれるが、実際の電子商取引では、それぞれの長所を取り入れて２種類の方式を組み合わせて使うことが一般的になっている。情報秘匿は、必要な情報を一見無関係なデータの中に埋め込むことで、流れるデータを傍受しただけでは特別な通信がなされていること自体が感知されないように図る技術である。２番目の「完全性」の実現方式としては、メッセージ・ダイジェストと秘密情報から作成した MAC (Message Authentication Code) で実現する秘密鍵型と、デジタル署名を利用する公開鍵型の 2 種類の方式がある。3 番目の「真正性」についてであるが、本人自身であることを保証する機能の実現技術としては、筆跡や指紋などのバイオメトリクスによる方式、固有の所有物による方式、秘密情報による方式の３種類がある。このうち秘密情報による方式では、パスワードや公開鍵暗号方式を用いたデジタル署名が一般的に用いられている。情報内容を保証する機能については、「完全性」の場合と同様になる。４番目の「責任追及性」の

電子商取引における典型例として、「否認防止」機能があるが、この機能の実現方式の一つとして公開鍵暗号方式を用いたデジタル署名がある。

このように、暗号技術は様々な形で使われており、本書はこれからの電子商取引分野の発展とともに暗号技術を取り入れたシステム構築を行っていく、システム事業者、システムインテグレータなどのシステムエンジニア、システムの運用に携わるシステム管理者などを主な読者とし、暗号技術を利用したセキュリティを実現するために必要な、暗号知識や暗号システムの構築方式について、含めて分かりやすく解説することを目的としている。

本書では２章において、セキュリティ基本機能における暗号利用と、暗号メールや SSL（Secure Socket Layer）や暗号化ファイルなどのセキュリティ応用機能における暗号利用とについて解説し、広く暗号技術の利用形態について述べている。また、最近重要性が高まっている知的所有権保護にて用いられる電子透かし技術の基本原理である、情報秘匿についても述べる。３章においては、共通鍵暗号、公開鍵暗号、ハッシュ関数、乱数、電子透かしなどの、暗号の仕組みと要素技術について述べる。これらを理解するには原理の理解が必要となるので、数学的理論なども一部交えて紹介するが、むしろこれらの技術は何ができるのか、個々の技術は何が異なりどのような弱点があるのか、などが重要である。また、今後どのような技術進歩や標準化がなされようとしているのかを理解していただきたいと考える。また、乱数は暗号の利用において非常に重要な役割を担っているため、様々な乱数についての特性や問題点を検討し、暗号技術での利用にあたって注意点を述べる。４章においては、暗号利用システムの要となる暗号鍵の管理方式について述べている。ここでは、鍵の生成、更新、破棄などの鍵のライフサイクルや、鍵の配送方法や保管方法などについて説明する。さらに、典型的な鍵管理システムや、標準化作業が進められている公開鍵基盤(PKIX)についても説明する。５章においては、暗号の安全性の面より暗号アルゴリズムの公開問題の重要性を説明し、暗号解読方法の側面から安全性の評価ついて解説する。さらに暗号の安全性評価に関連して、セキュリティ評価の国際標準であるＣＣ（Common Criteria）について紹介する。６章においては、暗号システムの構成方式や暗号ライブラリの最新の整備状況など、暗号利用システム構築において必要となる、暗号システムの実装・構築方式について述べる。また、暗号機能の検証、暗号利用とそのコストについても述べる。７章においては、国際社会での議論の方向を含めて、暗号の政策問題、各国や日本の政策の状況など、暗号に関する制度・法令について述べる。この中で、日本における暗号製品の輸出入に関わる各種法令の最近の状況は、暗号システム構築に際して注意すべき事柄である。

また、本書では詳しく述べていないが、その他の留意点として、知的所有権の問題がある。まず始めに、特許法、商標法などへの配慮をすることが重要である。最近はアルゴリズム特許（ソフトウェア特許）やビジネスモデル特許が成立するようになり、暗号アルゴリズムおよびその利用法の特許への抵触を注意する必要がある。さらに、特許期限が切れているものや、すでに特許申請されているが未成立であったり、状況が様々であるので、よく調査して世の中の特許状況を把握しておくことが重要である。また、国によっても特許制度が違ったり、ある特許については特許申請されている国が特定されていたりするので、各国の特許状況把握への配慮も重要である。さらに、商標登録についても同様な状況把握が大切である。２番目には、ソフトウェアの配布に関する問題がある。暗号以外のソ

フトウェアと同様に、著作権に配慮した利用・配布が必要である。さらに、暗号の安全保障に関連する各国の各種輸出規制や利用規制に抵触するおそれがあるので（7 章参照）、注意が必要である。

本書では、暗号技術の役割と利用形態、暗号の基本技術、暗号鍵管理方式などの暗号と暗号利用システムの基本的仕組みについて述べ、さらに暗号の安全性とその評価や、暗号システムの実装・構築方式についても述べており、暗号利用システムを構築または運用しようとする読者にとって、全般的な暗号関連技術の解説書となっている。しかしながら、暗号はあくまで「なまもの」なので、陳腐化しないように予め考えておくことが重要であるし、解読技術との追いかけっこである点には十分注意を払っていただきたい。また、ここで述べる各暗号技術も、いつ新しい攻撃方法が見つかるかもしれず、常に最新の情報を入手しながら、利用する暗号技術の評価を行う必要がある。さらに、より便利で使い易い暗号ライブラリや暗号ツールキットがでまわる可能性がある。また、暗号製品輸出規制に関連する制度・法令についても、規制緩和などの変化も予測されるので、世の中の動向に十分に注意することが必要である。本書もあくまで今現在の状況を述べるに過ぎないため、可能な物についてはインターネットの URL を記述してあるので、最新の情報も是非参照していただきたい。

2 暗号技術の利用形態

2.1 セキュリティ基本機能における暗号利用

まずここでは、セキュリティを実現するための基本機能である

1. 機密保持
2. 認証
3. 完全性保証
4. 否認防止

の各種概念の概要紹介と、そこでどのように暗号技術が利用されているかという点を概観する。

2.1.1 機密保護

機密保護とは、情報を正規の関係者以外の第三者にとっての有用性を失わせしめるための技術である。具体的には暗号技術と情報秘匿の二種類に分類できる。

2.1.1.1 暗号技術(Cryptography)

暗号技術というのは、つまるところ、可逆なデータ変換技術である。通信メッセージなど元となるデータ（**平文**と呼称）を一定の規則にしたがって前とはまったく異なるデータ（**暗号文**と呼称）に変換することで、その変換規則を知らない第三者には利用価値のないものとなす、というのがその基本といえる。平文を暗号文に変換する処理のことを**暗号化**、逆に暗号文を平文に変換する処理のことを**復号**[1]と呼ぶ。

可逆なデータ変換技術ということは、平文と暗号文との間は少なくとも一対一対応でなければならない[2]。このように一対一対応させるために可能なバリエーションとしては、平文が Nbit のデータである場合、$2^N!$ 通りのパターンがあり得ることになる（図 2-1）。

実は、暗号技術というのは、このように平文のサイズに応じて可能なすべての並び替えパターンを表として準備し、その表中の一つのパターンを選んで対応する暗号文を出力することに尽きてしまうとも言える。その際、常に同じパターンを利用し続けるのではなく、必要に応じてどのパターンを利用するのかをダイナミックに決めるためのパラメータが利用されるが、そのパラメータを指して『**鍵**』と呼んでいる。同じアルゴリズム（= 変換表をどのような順番で並べるか)であっても鍵を変化させることにより同一の平文に対する暗号文を異なったものとすることができる。このように、アルゴリズムそのものは公開してしまっても、鍵（後述の公開鍵暗号方式で言えば復号用の鍵のみということになるが）というパラメータさえ秘密に保っておけば安全性が確保できるというのが、現代的な暗号技術に望まれる性質である。

---

[1] “復号化”と表記されることがよくあるが、“復号”という単語自体に動作の意味がこめられているため、そこにさらに“化”という接尾語を付加するのは不適切であろう。この点、encryption/decryption と対称的に表現できる英語は便利である

[2] 平文対暗号文が一対多でも良い

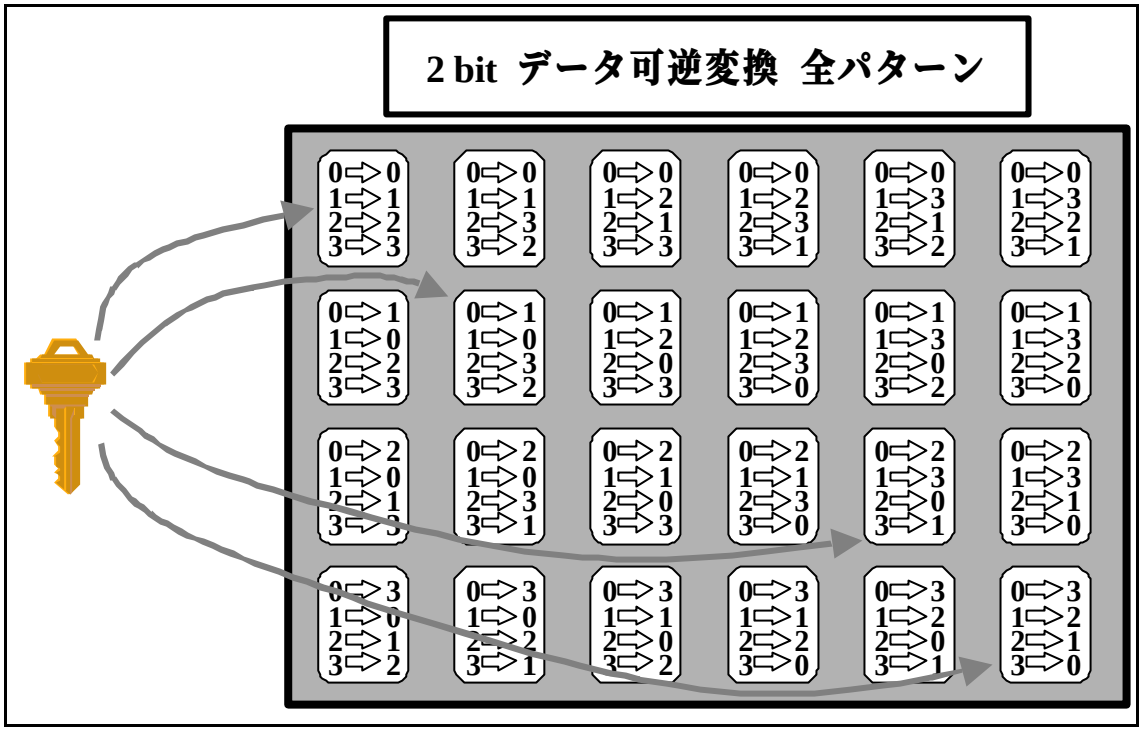


図 2-1　可能な可逆変換パターン

なお、常に同一値の鍵を用い続けるような方式では、解読される危険性が増すと考えられるので暗号化を多段で用いることが多い。そのときに用いられる鍵を作業鍵と呼ぶことがあり、一般的にはその時に発生した乱数（実際には疑似乱数）データを用いる。また作業鍵は後述の公開鍵暗号方式や、共通鍵を生成させる鍵管理アルゴリズムを用いて暗号化して、相手に配送する（作業鍵を暗号化する鍵をマスター鍵と呼ぶこともある）。図 2-2 に作業鍵の概念を示す。

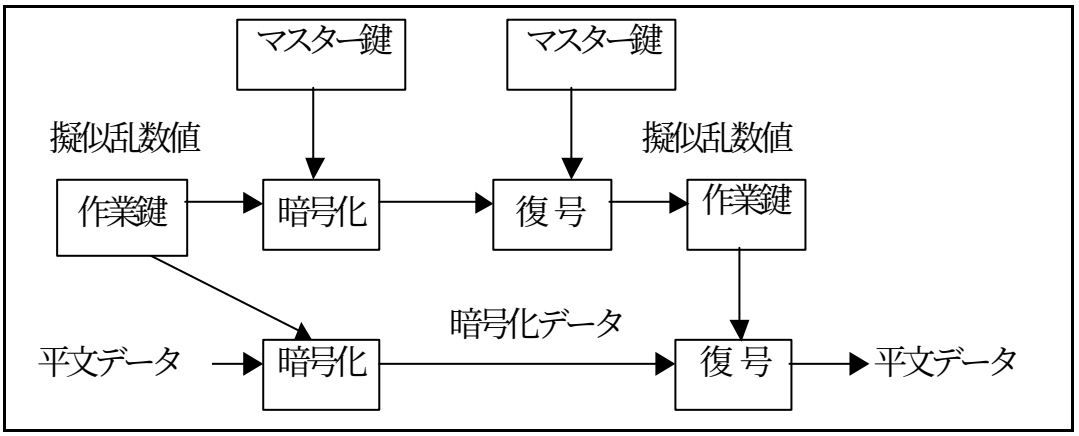


図 2-2 作業鍵

さて、上で暗号技術というのは並び替えのすべてのバリエーションを表にしたものに尽きる、と述べたわけだが、現実には、有名な DES [1]を始めとして多くの異なるアルゴリズムが利用されている。これは、ひとえに一定サイズ以上の平文に対してすべてのバリエーションを尽くすような変換表を用意することが現実的ではないということが理由である。

上で Nbit のデータに対して $2^N!$通りのパターンが必要と書いたが、この値は Nが６の時に 10^(89)というオーダとなる。これは全宇宙に存在する原子の数を超えてしまっている。つまり、全宇宙の原子に一個ずつ変換表が割り付けられたとしても、6bit という ASCII 一文字を表わすにも足りないデータに対する全変換パターンを尽くすことはできない、ということを意味するわけだ。

そのため、実際の暗号方式というのは、如何にして効率的に全変換パターンのサブセットを実現するのか、ということに重点を置いて、このような大量のデータを用いることなしで、かつ、推測されにくい変換を実現しようとするものと言うことができる。

実際の暗号方式は、共通鍵暗号（対称暗号・慣用暗号）と公開鍵暗号（非対称暗号）とに大別される。

(1) 共通鍵暗号（対称暗号・慣用暗号）

① 共通鍵暗号方式（Common-Key Cryptographic scheme）の原理

共通鍵暗号方式は、古来からの暗号と同じく、暗号化と復号に同一の鍵を用いる方式であり、公開鍵（非対称）暗号方式が発明されるまで、利用可能な暗号方式と言えば、この共通鍵暗号方式しかなかったため、慣用暗号とも呼ばれる。また、暗号化／復号アルゴリズムの内部処理、および鍵の作用から、対称暗号方式と呼ばれることもある。

共通鍵暗号方式では、暗号化と復号に同一の鍵を用いるため、この鍵を秘密にしておかなければ、通信や保管データの安全性が保てない。そのため、秘密鍵（Secret Key）暗号と呼ばれる事もあるが、公開鍵暗号で用いられるところの秘密鍵（Private Key：私用鍵ともいう）という用語との混乱を招く可能性があるため、その呼称は用いない方がよい。原理的には、平文ブロックの存在空間（bit 列の取りうる値）と暗号文ブロックの存在空間の写像が１対１の可逆変換であり、両方向の変換を行うためのマジックナンバー（鍵）として同じ値を用いるものである。

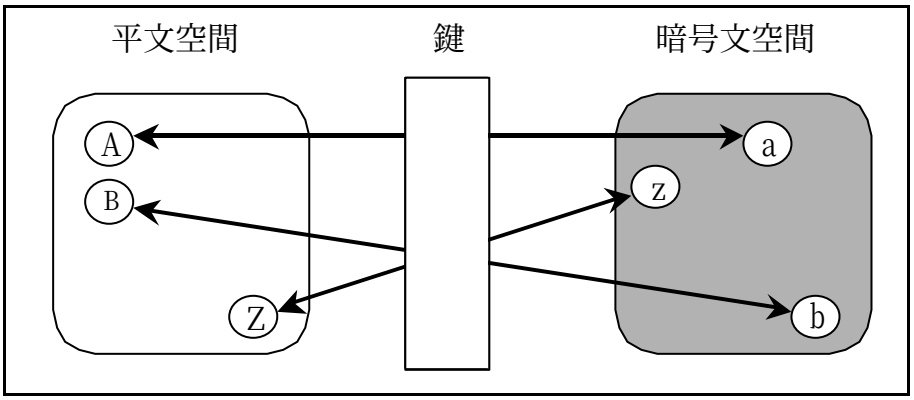


図 2-3 平文と暗号文の写像

通常は、復号処理は暗号化処理の逆の手順で行えばよい。

また、暗号の分類としては、平文データに対する操作の仕方により

1. ブロック暗号
2. ストリーム暗号（逐次暗号）

の 2 種類が存在する。

② 共通鍵暗号の長所と短所

1. 鍵はランダムに生成でき、その長さは比較的短くてよい。
2. 同一アルゴリズムであれば、解読の困難さは鍵空間のサイズ (鍵長が N bit ならば、2^N) に比例する。
3. アルゴリズムにより、線形解読法、差分解読法などの手法を工夫すればすべての鍵を試行しなくとも解読が可能になる場合が存在する。たとえば、後述の DES では、線形解読法という手法を用いることにより、既知の平文に対して計算量が 12bit 分の１程度に少なくなることが知られている。
4. 鍵が共有されるので、ある暗号文を生成したのが鍵所有者のうちの誰であるかまでは特定できない。
5. 一度鍵が解読されると鍵を共有する多くの利用者が脅威にさらされるため、鍵の生成、配送、保管に十分注意する必要がある。
6. 暗号強度の根拠は鍵の長さと演算結果として得られる出力のランダム性のみに依存するため、計算量が少ない（高速である）。
7. 同様に専用ハードウェアの開発が比較的簡単であり低コストである。
8. すべての鍵を試すことにより正しい暗号化／復号鍵が決定できる（総当り法）、当然、無条件安全性 (後述) は保証され得ず、また、計算量的安全性 (後述) の保証も現状は経験的に言えるのみである。
9. 鍵は１つの利用主体もしくは通信経路毎に必要であるため、ｎ人の利用者と暗号通信する場合、ｎ個の異なる共通鍵を秘密に保持する必要がある。

③ 共通鍵暗号アルゴリズムの適用分野

共通鍵暗号アルゴリズムは上記のような特長があり、電子署名など原理的に公開鍵暗号でなければならない場合を除き、ほとんどすべての暗号の応用領域で利用される。

特に、大量のデータを暗号化する際は、必ず用いられている。

不特定多数との暗号通信のように、共通鍵暗号でなく公開鍵暗号を使った方が良い場合も、通常は公開鍵を用いて共通鍵（= 前述の作業鍵）の交換を行ない、データの暗号化そのものは共通鍵暗号アルゴリズムが用いられる。この場合、必要になる度に鍵を交換しても良いため、鍵の保管における安全性配慮や、鍵の管理についての負担が軽減されるというメリットも生じる。

ただし、同じ鍵を用いて同一のデータを暗号化した場合、常に同一の結果が出力されるため、単純に共通鍵暗号を用いたプロトコルの設計を行うと、攻撃者が過去に盗聴しておいた正規ユーザ間の通信を再利用することで、正当な通信相手になりすますというタイプの攻撃法 (リプレイ攻撃) が実現できてしまう可能性がある。このような攻撃を防ぐためには、データが毎回異なるよう、暗号を利用するシステム側で配慮されなければならない。また、暗号アルゴリズムのレベルでみた場合でも、ブロック暗号を用いる際には後述の CBC (Cipher Block Chaining) モード等、データを連鎖させることで同一内容の平文ブロックが異なる内容の暗号文ブロックに変換されるように考慮されている方式を用いることが重要である。

(2) 公開鍵暗号（非対称暗号）

① 公開鍵暗号（非対称暗号）とは

公開鍵暗号（非対称暗号）とは、暗号化と復号に異なる鍵を用いる暗号である。いずれか一方の鍵から他方の鍵が容易に計算できないため、一方の鍵を公開することができるという特徴を有する。他方は秘密に保持する。

② 公開鍵暗号方式（Public-Key Cryptographic scheme）の原理

公開鍵暗号では、利用者Aが利用者Bに暗号通信する場合以下の手順で行う。

[鍵生成]各利用者は、あらかじめ一対の鍵を生成し、一方（公開鍵（Public key））を公開し、他方（秘密鍵（Private Key））を秘密に保持する。

[暗号化]利用者Aは、利用者Bの公開鍵を用いて平文を暗号化し、利用者Bに送信する。

[復　号]利用者Bは、自分の秘密鍵を用いて受信した暗号文を復号する。

このとき公開鍵暗号は、以下の２条件を満たすように構成される。

1. 暗号化と復号の計算は比較的容易である。
2. 暗号解読（Bの秘密鍵を知らない者が、Bの公開鍵、暗号文、および暗号アルゴリズムから元の平文を計算すること）は非常に困難である。

③ 公開鍵暗号の特徴（長所と短所）

公開鍵暗号は、共通鍵暗号と比較して以下の長所・短所を有する（表 2-1参照）。

A. 鍵の秘密配送が不要

共通鍵暗号では、暗号通信に先だって秘密に共通鍵を配送（共有）する必要があるが、公開鍵暗号では、公開されている相手の公開鍵を入手するだけでよい。

B. 秘密に保持する鍵が少ない

ｎ人の利用者と暗号通信する場合、共通鍵暗号では、ｎ個の異なる共通鍵を秘密に保持する必要があるが、公開鍵暗号では、自分の秘密鍵を保持するだけでよい。

C. 電子署名（デジタル署名）が実現できる

共通鍵暗号では､ある共通鍵で暗号化された文書がその共通鍵を共有する当事者のいずれによって暗号化されたかが特定できないが、公開鍵暗号では、ある秘密鍵で暗号化された文書はその秘密鍵を保持する当人しか暗号化できないため誰が作成したかが特定できる。

D. 秘密鍵の保管の重要性

特に電子署名に利用される場合の秘密鍵は､後述の否認防止効果を期待されることが多いため、これを本人のみが秘密に保持することが非常に重要となる。

E. 共通鍵暗号に比べて処理が低速

公開鍵暗号では、共通鍵暗号に比較して、暗号処理が低速である。このため、多量のデータの暗号通信には共通鍵暗号を利用し､共通鍵暗号に用いられる共通鍵の秘密配送と電子署名に公開鍵暗号を利用するのが一般的である。

F. 公開鍵の正当性確保が必要

公開鍵暗号では、公開鍵を公開できる反面、それが偽の公開鍵にすり変えられると大きな問題となる。これを防ぐため、公開鍵が誰の公開鍵であるかを証明する機関（認証局(CA：Certification Authority)）を利用するのが一般的である。

G. 暗号通信による相手認証は不可能（匿名暗号通信）

共通鍵暗号では、当事者しか知らない共通鍵を用いて暗号通信するため、暗号文が正しく復号できるかどうかによって通信相手を認証できる。しかし公開鍵暗号では公開鍵を用いて誰でも暗号通信できるため、受け手は誰から暗号文が送られてきたかが分からない。言い換えれば匿名の暗号通信が可能（なお公開鍵暗号の電子署名機能を用いれば、相手認証が可能である）。

表 2-1 公開鍵暗号と共通鍵暗号の比較

	公開鍵暗号	共通鍵暗号
a)鍵の秘密配送	○不要	×必要
b)秘密に保持すべき鍵	○自分の秘密鍵のみ	×通信相手毎に必要
c)電子署名	○可能	×困難
d)処理速度	×低速	○高速
e)公開鍵の管理	×必要	○不要
f)暗号通信による認証	×困難	○可能

2.1.1.2 情報秘匿(Steganography)

情報秘匿は、必要な情報を一見無関係なデータの中に埋め込むことで、流れるデータを傍受しただけでは特別な通信がなされていること自体が感知されないように図る技術である。たぶん日本でもっとも良く知られた情報秘匿の例としては、『いろは歌』が挙げられるだろう。

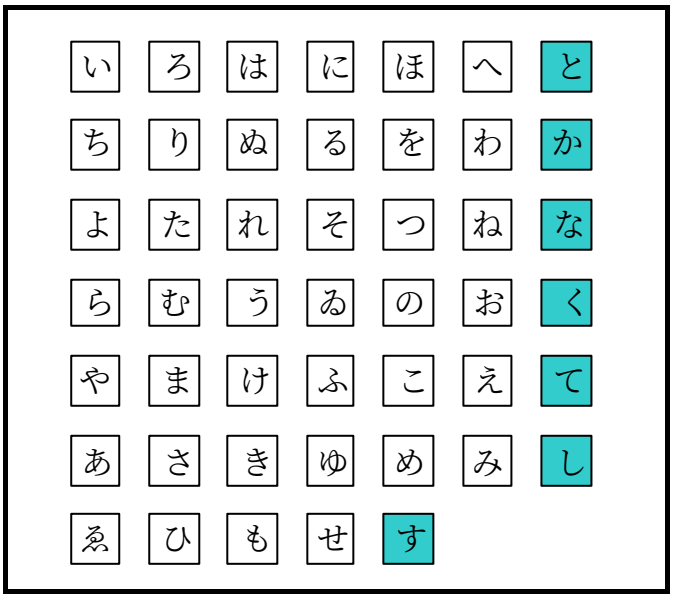


図 2-4 いろは歌の情報秘匿

一見した限りでは『色は匂えど散りぬるを我が世誰ぞ常ならん有為の奥山今日越えて浅き夢みし酔いもせず』という歌だが、これを 7 文字ずつ (最終ブロックだけは 5 文字) に分けて各ブロックの最終文字だけを読み下せば『とかなくてしす』(咎なくて死す) という文章が浮き上がる。これは一説に柿本人麻呂の作とも言われるそうだが。

情報秘匿が持つ特徴の一つとして、通信文のサイズが本来の目的である情報が必要とす

るよりも大きなものとなってしまう、という点がある。たとえば、上記いろは歌の例では、必要な７文字の情報を伝えるのに 47 文字が費やされているが、このような冗長性は本質的に必要とされるわけだ。

デジタル情報に関して端的な情報秘匿の使用例としては、画像データに他の情報を埋め込む（implanting)という手法が知られている。フルカラーの画像データは１画素あたり R/G/B　３色×８bit、計 24bit で表わされるわけだが、たとえば R/G/B それぞれに相当するデータの最下位１bit のみを本来の画像データとは無関係な他のメッセージを表わすために使う、という方式だ。幸い、ほとんどの人間の目というのは $1/2^8$ というような微妙な色の違いを判別できるようには作られていないので、このように乱暴なことをやってしまっても表示される画像の質にはほとんど影響がない。この方法では 24bit 中の３bit だけが本来目的とするデータを表現するために利用されるわけだが、たとえば VGA サイズの画像データの場合でも、640×480×３÷８= 115.2KB と、結構な量のデータが表現できることになる。

データがランダムに近い bit 列に変換されてしまうためにかえって重要な通信が行われている事実そのものは傍受者に推測されてしまう暗号技術とは異なり、情報秘匿では一見無価値なデータの中に重要なデータを隠すことができるため、傍受者の注意を惹きつけ難いというメリットがある。半面、アルゴリズム（あるデータをどのようにして他のデータに埋め込むか）が漏れてしまうと容易に本来のデータを復元されてしまうというデメリットもある。そのため、情報秘匿単独ではなく、本来のデータを暗号化した上で他のデータに埋め込む、といった暗号技術とのハイブリッドな利用が望ましいといえるだろう。

近年、情報秘匿は、デジタル・コンテンツに対して作成者に関する情報を埋め込んでおくことで、知的所有権保護策として利用されることが多い。このような利用方法を『電子透かし』と呼ぶ。ただし、電子透かしとして利用する場合に、たとえば前出のような画像データの各画素データ中 R/G/B 最下位１bit を利用して情報を埋め込むという類の誰でも容易に検出できるような手法を採用するのでは、これまた誰でも容易にその『透かし』を外すことができてしまうため、上手く機能しない。そのため、実際の用に足る電子透かしを実現しようとして、３章で紹介するように多くの方式が提案されている。

このような電子透かし方式の中には、仮想的にアナログ化、つまり、たとえば画像データの場合で言えば、データを一回プリントアウトし、それを再度スキャナで読み込む、といったプロセスをシミュレートした後でも透かしデータが外れないような方式も存在する。

しかし、一般的な話として透かしを外しにくくすればするほど、デジタル・コンテンツの質（画質、音質など）に悪影響を与える度合いも高まる傾向にあるという難点もある。また、知的所有権保護という観点から考えると、正規のユーザによる不正コピーを防止する必要があるわけだが、この場合の正規ユーザはデジタル・コンテンツの知的所有権保護ということとは直接利害関係を持たないため、それほど熱心に護るだろうことはあまり期待できない (というより、意図的に不正を行うことすら容易に予想できる)。そのため、電子透かし技術というのは、正規のユーザによる不正にも耐え得るようなものでなければならないが、これはなかなか実現するのが難しい技術であろう。ちなみに、通常の暗号技術の場合は、自分の秘密鍵を護ることが本人にとって直接の利益となるため、このような厄介な問題とは無縁であり、したがって実用的なシステムの実現が比較的容易であったとも言えるのかもしれない。

2.1.2 認証

認証とは、ある存在が真正のものであることを確認する行為である。人間などの物理的な存在の実在性を確認するための本人認証と、データなど情報の真正性を確認するためのデータ認証とに大別できるが、後者に関しては次節の完全性保証で紹介することにして、ここでは前者の本人認証に話を絞ることにする。

ECOM フェーズ１における「本人認証技術検討WG」の報告書である『本人認証技術の評価基準(第 1 版)』[2]では、このような本人認証手段として

1. バイオメトリクス
   (ア) 指紋・虹彩パターンなど
   (イ) 声・筆跡など
2. 所有物
   (ア) IC カードなど
3. 秘密情報
   (ア) パスワード、暗証番号など
   (イ) ある種のワンタイムパスワードシステム、公開鍵暗号技術など

のように分類している。

このうち、バイオメトリクスというのは認証される主体のみが生来的に備える情報のことで、これはさらに指紋のように決して他者が模倣できないものと、筆跡のようにそれなりの技術を持った者ならばある程度の模倣が可能なものとに分類されている。

所有物というのは、本人以外持ち得ない物品を利用するものである。

最後の秘密情報とは、本人（と、必要な場合は認証側）のみが知る情報を認証の手段とするもので、これはパスワードのように本人と認証側が秘密情報を共有しなければならないものと、公開鍵暗号技術のように真に秘密の情報は本人のみが所有していればよいという類のものに分類できる。

以上のような情報を認証側にあらかじめ登録しておき、実際の場面では、被認証側主体から提示される情報を登録情報と突き合せることによって可否を判断する、というのが本人認証行為であると定義することができるだろう。

ただし、バイオメトリクスもしくは所有物を単純に利用するという方式の場合、認証時点で提示される情報を奪取されることでリプレイ攻撃の手段とされる可能性があるため、インターネットのようなネットワークを介した形で認証を行う場面では適さないことが多い。そのため、これらはたとえば入退室管理のように、指紋スキャナのようなデバイスの前に本人の物理的な出頭を求める場面での認証手段となることが主である。

ことネットワークを介した認証ということでは、以下に述べるように何らかの形で秘密情報を利用したものが当面は主流となるだろう。

2.1.2.1 電子署名

公開鍵暗号技術の重要な応用の一つに電子署名（デジタル署名）がある。『認証』という行為を実現する場合、この技術が非常に重要となるため、まず、その説明から開始しよう。

リアル社会では通常、公的な文書には捺印や署名をすることで、その文書の作者の身元を明らかなものとしているが、バーチャル社会で電子的な情報に対して同様の機能を実現

することを可能たらしめるのが電子署名という技術である。

電子署名は、さらに署名処理後にデータに対して改変が加えられたかどうかという点まで明らかにすることができるため、改竄防止という観点から見れば、リアル社会における印鑑押捺／署名よりもさらに強い機能を持ったものと捕らえることも可能である。

一般に、電子署名は一方向ハッシュ関数と、公開鍵暗号技術とを併用することによって実現される。

(1) 一方向ハッシュ関数

一方向ハッシュ関数の詳細に関しては 3.3 章で紹介するが、ここで簡単に説明しておけば、これはハッシュ関数 (任意サイズのデータを固定長のデータ (ハッシュ値) へとマッピングする関数) のうち、

1. 与えられたハッシュ値が得られるようなデータを求めることが困難である (一方向性)
2. 同じハッシュ値を得られるような複数の異なるデータを求めることが困難である (非衝突一致性)

という二種類の性質を満たすような類のことである。

このような一方向ハッシュ関数を用いて得られたハッシュ値というのは、元となるデータの『指紋』、つまり、そのデータを特徴的に示すものとして用いることが可能になる。

(2) 電子署名の原理

一般的に電子署名には、公開鍵暗号を使用し、以下の手順で確認や偽造、不正のチェックが行われる。

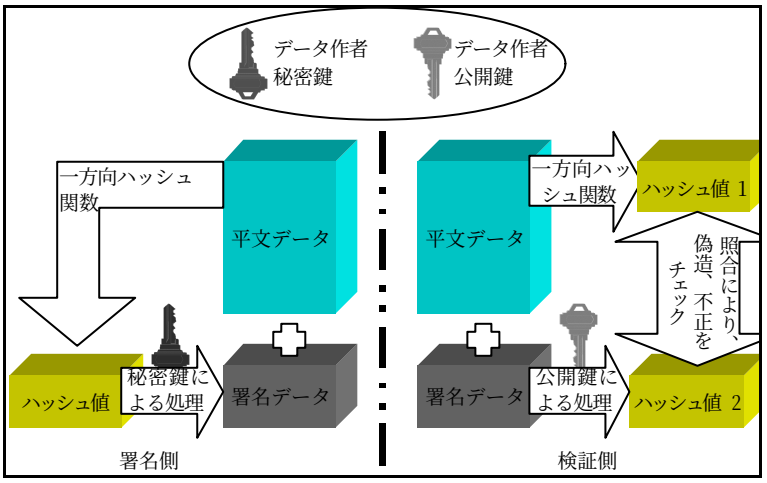


図 2-5 基本的な電子署名の手順

2.1.2.2 秘密情報を利用する認証手法

(1) パスワード

パスワードは、認証側／被認証側が同じ秘密の情報であるパスワードを共有し、これを利用して認証を行う、というものだ。基本的には、正しくパスワードを言える (= タイプできる) 主体が本人として認証されることになる。

① 単純パスワード

単純パスワード方式は、文字通り単純にパスワードそのものを認証側のデータベースに蓄えておいて、認証時には被認証側から提示されるパスワードをそのままデータベース中の値と比較することで、本人かどうかを判断する。

認証時のオーバヘッドは少ないが、半面、万一認証側システムに侵入されてパスワード・データベースが漏洩した場合、データベース中に存在するすべてのユーザのパスワードが即座に危殆化することに繋がる。

② ハッシュ式パスワード

ハッシュ式パスワードは、ユーザのパスワードを一定の方式 (= 一方向ハッシュ関数) で変換し、その変換されたデータ (ハッシュ値) のみを認証側のデータベースに蓄える方式のことである。認証時には、被認証側から提示されるパスワードを登録時に利用したのと同じ一方向ハッシュ関数を用いて変換した上で、データベース中の値と比較する。

この方式の場合、パスワード・データベースの漏洩が起こった場合でも、ただちにユーザのパスワードの危殆化には繋がらない。なぜならば、一方向ハッシュ関数の性質から、データベース中のデータに合致するようなハッシュ値が得られる被ハッシュ・データを求めることが、一般には非常に困難だからである。ただし、ユーザがパスワードとして選びやすい文字列を辞書として用意し、それらに対して順番に一方向ハッシュ関数を適用することで、そのうちのどれかからデータベース中のデータと合致するハッシュ値が得られることを期待する、というタイプのいわゆる『辞書攻撃』と呼ばれる手法が存在するため、ハッシュ式パスワードの場合でも、パスワード・データベースの安全な保管には十分注意する必要がある。

ただし、ハッシュ式パスワードを用いたとしても、ネットワークを経由して認証を行う場合には、パスワードそのものが流されることになるため、リプレイ攻撃の危険を免れることができない。

ネットワークを介した安全な認証処理を実現するために、以下に述べるようないくつかの手法が知られている。

(2) チャレンジ&レスポンス

チャレンジ＆レスポンス方式は、ユーザ (=被認証側) を認証する際に、システム (認証側) がまずランダムに生成したデータ (= チャレンジ) をユーザに送り、ユーザはそのチャレンジと秘密データ (= パスワード) とを結合したデータに対して一方向ハッシュ関数を適用し、その結果をシステムに送り返す (= レスポンス)というものだ。

この方式の場合も、通常のパスワード方式と同様、パスワード情報はユーザとシステムとが共有しているため、システムでも独自にチャレンジとユーザのパスワードと

から同じ演算を行ない、その結果とユーザから送り返されたレスポンスとを比較し、両者が一致すれば正しいユーザということになる。

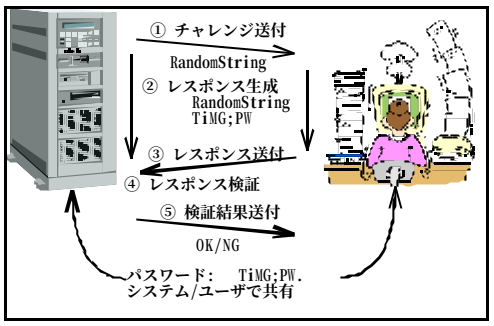


図 2-6 チャレンジ & レスポンス方式

この方式の場合、システムとユーザとの間でやり取りされるのは、チャレンジというランダムなデータと、そのチャレンジとパスワードとを結合したものに対して一方向ハッシュ関数を適用したデータという二つだけであり、一方向ハッシュ関数の性質から、レスポンスからユーザのパスワードを導き出すのは非常に困難である。そのため、システムがチャレンジを正しく一回限りのランダムなデータとしてさえいれば、ネットワークを介して認証を行なったとしても、それらのデータを用いることによってユーザになりすますことは不可能である。

この種の認証を行うシステムとしては、メーラがサーバからメールを取得するためのプロトコルである POP (Post Office Protocol)[3] の認証手段の一つである APOP 方式や、電話回線などを通してインターネット接続を行うためのプロトコルである PPP(Point-to-Point Protocol) [4] における CHAP(Challenge Handshake Authentication Protocol)[5] 方式などが知られている。

(3) ワンタイム・パスワード

ワンタイム・パスワードという方式は、認証のために用いられるパスワードデータを文字通り一回限り (onetime) しか通用しないものとすることで、たとえそのデータがネットワークを介して通信されるところを盗聴されたとしても、そのデータを用いてリプレイ攻撃することはできないように設計されたシステムである。

ワンタイム・パスワードの具体的な実現手法としては何種類かが存在する。前項のチャレンジ&レスポンス方式もワンタイム・パスワードに分類されることもある。また、システム側と同期して動的にパスワードを生成する特別なハードウェアを用いるものもある。さらに、S/Key や OPIE (Onetime Password In Everything) などと呼ばれる完全にソフトウェアのみで実現される方式もある。

最後の S/Key もしくは OPIE というのは、組版システム『LaTeX』の生みの親としても知られる Lesrie Lamport が発表した論文[6]中のアイデアを実装したもので、本当の秘密データは認証側にさえ知らせる必要がなく、被認証側のみが秘密に保持しておけば済むような形となっており、前述の ECOM 『本人認証技術の評価基準(第 1 版)』では『秘密情報』の二番目に分類される方式である。

(4) 公開鍵暗号方式

前述のとおり、電子署名はデータに対してその作成者が誰であるのかということを、確かな形で記すための手法である。ということは、電子署名という機構がそのままユーザ認証として使えるということが明らかであろう。たとえば、図 2-7のような方式で電子署名を利用した認証を実現することができる。

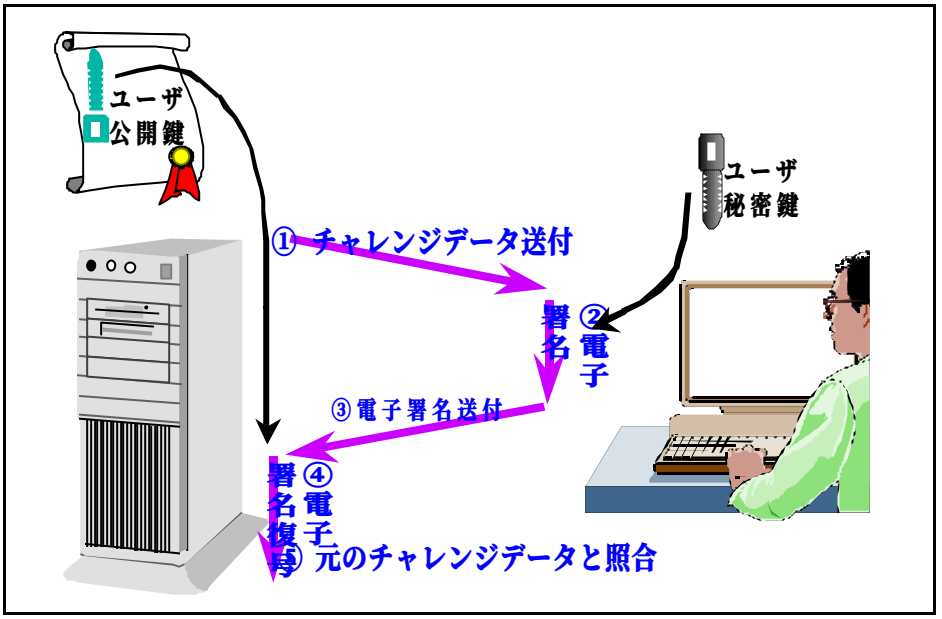


図 2-7 公開鍵暗号方式を利用した認証

この方式では、前出のチャレンジ＆レスポンス方式と同様、システム側が毎回異なるランダムなチャレンジを送るように取り計らう限り、安全な認証処理が実現できる。なぜならば、公開鍵で正しく復号できるような形でチャレンジデータを処理できるのは、対応する秘密鍵を持つユーザ本人に限られるからであり、チャレンジデータが正しく一回限りのものとなっていれば、電子署名データを送る際に盗聴されたとしても、後の時点でそのデータを再利用するリプレイ攻撃は行なえないからである。

ただし、上図に示した方式をそのまま単純に利用すると、ユーザにとっては危険なこととなり得るので注意が必要である。システム側に悪意が存在した場合、任意のデータ[3]に対してユーザに署名を行なわせることが可能であるから。

そのため、実際に運用するシステムでは、チャレンジデータの選択に関して認証を行う側のみが制御権を持つようなものであってはならない。

2.1.3 完全性保証

完全性保証は、前述のデータ認証と同じ意味であり、あるデータが作成後に改変されていないことを保証する手段である。完全性保証手法としては、秘密情報と一方向ハッシュ関数による共通鍵型と、電子署名を利用する公開鍵型の２種類に大別できる。

2.1.3.1 秘密鍵型の完全性保証

秘密鍵型の完全性保証システムとは、データ作成側と検証側とが秘密に共有する情報 (= 共通鍵) およびデータそのものとから MAC (Message Authentication Code: メッセージ認証子) を生成し、この MAC をデータに付加することでデータ完全性保証を実現する仕組のことである。

一般に MAC は、図 2-8のように、データ本体と秘密情報とをなんらかの方法で結合したデータ全体に対して一方向ハッシュ関数を適用することで生成される。ハッシュ関数の一方向性により、MAC から元の秘密情報を復元される危険性は極端に低い。

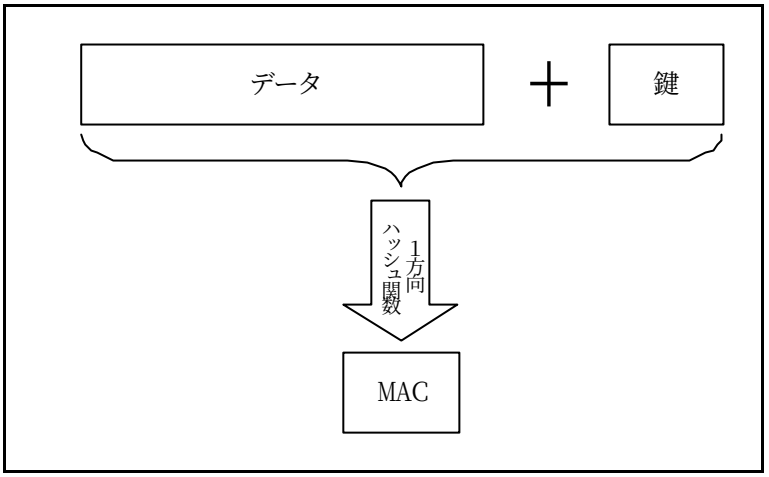


図 2-8 MAC の原理

MAC 生成方式としては、データ本体と秘密情報を単純に結合するというシンプルなものから、HMAC[7] と呼ばれる方式のようにある程度の安全性が証明されているものまで、さまざまなものが存在する。当面のトレンドとしては、HMAC のように証明による裏付けを持った方式が好まれるだろう。

SSL や IPSec などのように形成したコネクションを流れるデータをダイナミックに扱う必要があるような場面では、一般に、この共通鍵型のデータ完全性保証方式が用いられる。

2.1.3.2 公開鍵型の完全性保証

前述のとおり、電子署名は一方向ハッシュ関数の持つ一方向性および非衝突一致性により、本質的にデータ完全性保証機構を備えている。そのため、電子署名を施したデータというのは、そのまま完全性保証をも実現していると言える。

ただし、電子署名は公開鍵暗号技術を利用するために処理速度は決して速いものではないため、たとえば電子メールなどのように署名処理に要する遅延がそれほど影響しない場面でのみ利用される。

前項に記したとおり、コネクション・レベル等でリアルタイム性が要求されるような場面では、処理速度の速い秘密鍵型のデータ完全性保証機構が用いられることが多い。

---

[3] ユーザにとって不利益を被るようなデータでもあり得る

2.1.4 否認防止

否認防止は、ある主体が過去に自分の行った行為をなかったものと主張することを不可能にせしめる機能である。ネットワークを介した取引などを行う場合には、この機能を実現する必要がある。

通信行為における否認には送信者による否認と受信者による否認の 2 パターンが考えられるが、このうち送信者による否認はデジタル署名により防止することができる。なんとなれば、公開鍵暗号方式によるデジタル署名の場合、署名を作成するためには作成者の秘密鍵を用いる必要があるが、秘密鍵を利用できるのは唯一その所持者のみであり他の誰とも共有はされない性格のものであるため、あるデータに対してデジタル署名が施されているということは、すなわち、そのデータの作者が一意に決まるということを意味するからだ。これが、他の主体と何らかの形で秘密情報を共有しなければならない他の手法の場合には、その共有主体から秘密情報が洩れたのだと言い張ることにより、過去の行為の否認が可能になる。

さらに進んで受信者による否認まで防止することを望む場合には、事象の発生したことを証明してくれるいわゆる公証サービスの導入が必要となるだろう。

2.2 セキュリティ応用機能における暗号利用

それでは、実際の応用としてどのように暗号が利用されているのだろうか。本節では、さまざまな場面で暗号技術が利用されている部分を概観してみる。

2.2.1 利用形態による分類

まず、暗号技術が利用される場面を、その利用形態によって大きく分類してみると、以下のようになるだろう。

表 2-2 暗号技術利用形態

アプリケーション型	電子メールのように、ユーザが直接利用するアプリケーションが暗号技術をサポートするケース
コネクション型	通信路でデータの暗号化などが行われるケース。典型的な例としては SSL が挙げられる。コネクション型はさらに、PC ⇔サーバ間等のエンドポイント同士で処理する end-to-end 型と、ルータ間等の中継機器間毎に暗号化／復号などの処理を行う link 型とに分類されることがあるが、本節では特に両者を区別しなくても済むレベルの説明にとどめる
ストレージ型	ハードディスクなどに格納されるデータに対して暗号化を行うケース

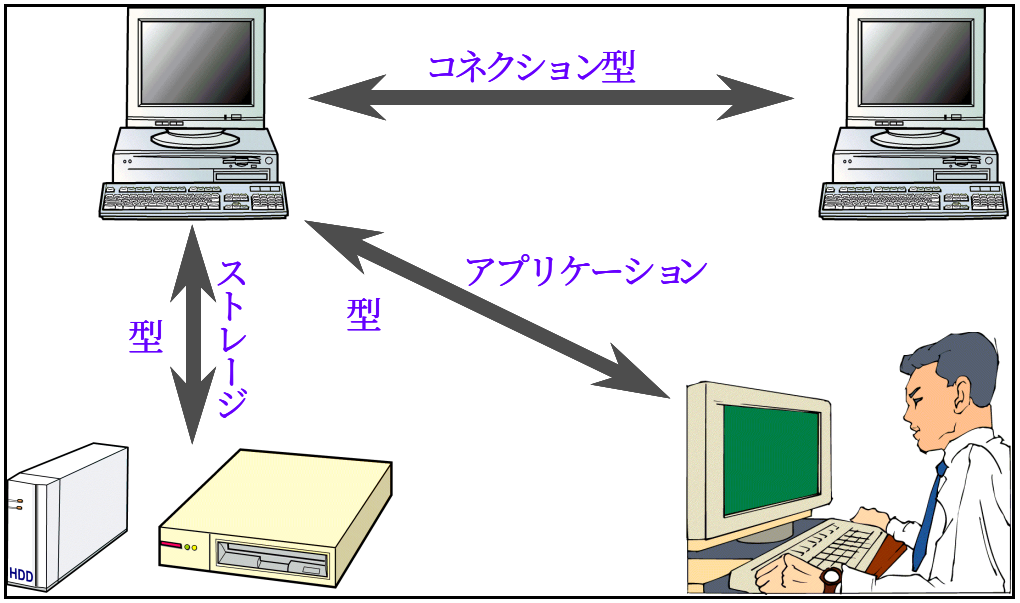


図 2-9 暗号技術利用形態

以下では、それぞれの形態における利用例に関して簡単に説明しようと思う。

2.2.2 アプリケーション型

2.2.2.1 セキュア・メール

セキュア・メールは、送信者が発信するメール・メッセージに対して、デジタル署名・暗号化などの保護策を適用するものである。典型的には、暗号化などの処理は送信者がメッセージを発信する時点で実施され、そのままの形で受信者のメール・ボックスにまで届き、受信者が読む時点で復号などの処理が行われる。そのため、通信途中はもちろん、メール・ボックスに格納された状態でもメッセージに対する保護は有効となる。

メールという非常に重要なコミュニケーション・ツールに対する保護策となるため、セキュア・メールに関しては、独自規格を利用するアプリケーションまで含めると、多数の製品が存在するが、ここではそのなかでも特に普及が進んでいる二つの規格である S/MIME と PGP を紹介しようと思う。

(1) S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extensions)[8] は、RSA Security 社が提唱し、現在インターネットにおける標準化活動の過程に置かれた規格である。特徴として、公開鍵暗号技術利用アプリケーション相互の運用性を向上させるための手段として RSA Security 社が提唱する PKCS (Public Key Cryptography Standards) [9] に基づく暗号化・デジタル署名などの機能を、電子メールに対して適用可能としたものということができるだろう。

また、“S/MIME”という名前が示す通り、インターネット・メールの標準的規格である“MIME”[10]に準拠したものとなっている。

動作の特徴は、メッセージの暗号化に関しては、PKCS で『Digital Envelope (デジ

タル・エンベロープ)』と呼ばれる方式、つまり、送信者によって勝手に作成されたランダムな鍵 (= セッション鍵) を用いてメッセージを共通鍵暗号方式にて暗号化した後に、当該セッション鍵を受信者の公開鍵で暗号化した上でメッセージに添付する、という方式が採用されている。

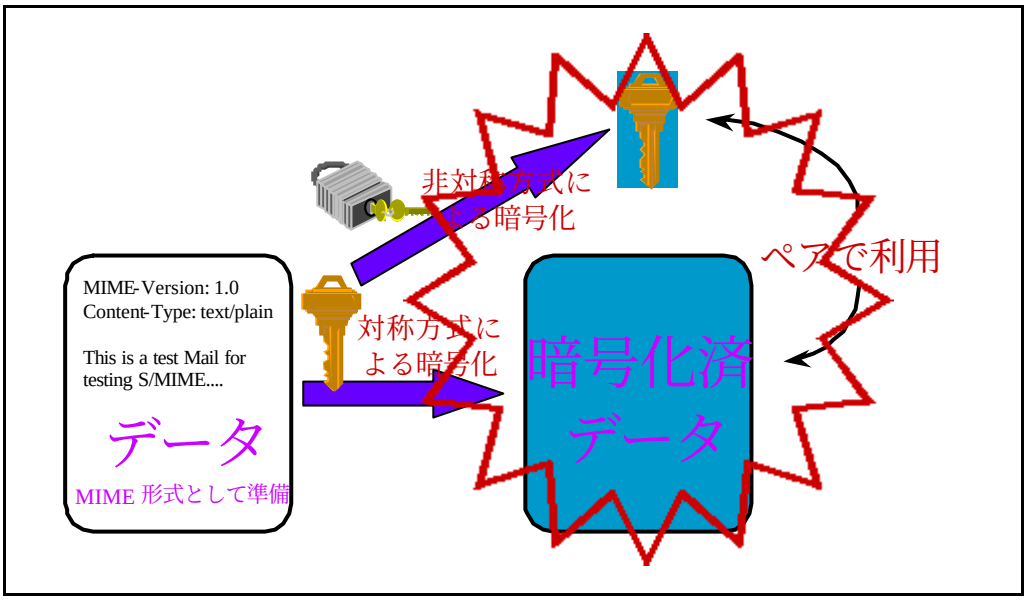


図 2-10 S/MIME による暗号化メール構成法

また、デジタル署名を施したメッセージとしては、PKCS 独自形式と、MIME が定めるところの“Security Multiparts for MIME”と呼ばれる形式の二種類が用意されている。この二種類の形式には、表 2-3のように一長一短があるため、適宜使い分ける必要がある。

表 2-3 S/MIME 署名メッセージの形式による差異

	長所	短所
PKCS 独自形式	転送途中でメッセージ本文に対してサーバによる改変を受ける (= デジタル署名の検証が行えなくなる) 可能性が、できる限り低く抑えられている	PKCS を解釈できないメーラを利用する場合、署名が検証できないだけでなく、メッセージ本文すら一切読むことができない
Security Multiparts for MIME	PKCS を解釈できないメーラを利用する場合でも、メッセージ本文を読むことだけは可能	メッセージ本文がオリジナルのまま添付されるため、古いメール・サーバなどでは改変を受ける (文字数の多い行に対して途中で改行を入れられる、等) 可能性がある (= デジタル署名の検証が行なくなる)

Netscape Messenger および Microsoft Outlook {Express,98,2000} 等のメジャーなメーラが最初からサポートしていることもあり、また、認証機関が発行するデジタ

ル証明書の利用が必須であるため相手の身元認証が厳密に行えるという利点もあって、特にビジネス場面では頻繁に利用されるようになっている。

(2) PGP

PGP (Pretty Good Privacy)は、もともと米国の Philip Zimmermann 氏が開発した暗号化アプリケーションであり、インターネット等で無償で公開されていたものだ。現在では Network Associates 社から商用版も提供されているし、GPG (GNU Privacy Guard) と呼ばれる独自開発の互換フリーソフトウェアも存在する。

暗号化などの処理を施されるため一般的なバイナリ形式となってしまうデータを、電子メール（SMTP プロトコル）で運べる形式に符号化する機能など、特にメール・メッセージを処理するのに便利な仕掛けを備えているため、セキュア・メールを実現する手段として利用されることが多いが、CERT (Computer Emergency Response Team)[12] が発行する勧告に対して偽造防止のための署名を行うなどの用途にも役立てられている。

メール・メッセージに対して暗号化・デジタル署名などを行う基本的な枠組は、ほぼ S/MIME と同様であるが、現状では両者の間で互換性はない。ただし、スタート時点は特定のアプリケーションだった PGP も “Open PGP”[11]としてインターネットでの標準化の過程にあるため、今後は両者が相互運用可能になる期待もある。

また、“Web of Trust”と呼ばれるユーザ同士が個人的に信頼を与え合うというスキームに基づいているためにデジタル証明書を発行する認証機関の存在が必要とされないため、手軽に利用できるという特徴を持つ。

2.2.2.2 クレジット決済

クレジット決済を行うスキームとしては、Visa と MasterCard が共同で開発した SET (Secure Electronic Transaction) が有名である。

SET は、消費者 (Card Holder), 小売業者 (Merchant), そして決済処理を行うための Peyment Gateway という三者の間でデジタル署名／暗号化などで保護されたトランザクションを行うことで、インターネットのように開放的なネットワークを介した場合でも安全に取引が行えるように考案された仕組である。

技術的には、

- 暗号化用／署名用に別々の鍵ペアを利用する所謂『Dual Key』型であること (ただし、消費者は署名用の鍵ペアのみがあればよい)
- 消費者からのリクエストが、小売業者と Payment Gateway それぞれに対して必要な情報のみを提示するだけで済むように、『Dual Signature』と呼ばれる仕組を提供していること

といった点が興味深いと言えるだろう。

2.2.2.3 ソフトウェア署名

ここで言うソフトウェア署名とは、実行形式のオブジェクト・データに対して、その作者を特定し、かつ製造後改変が加えられていないことを保証することができるようにデジタル署名を施したものを指している。

小売店で販売されるソフトウェアの場合は、パッケージに印刷された会社名や包装用セロハンなどが破損していないことを確認することで同じ効果が得られるため、このような機能は必要とされないだろう。そのため、この機能は主としてインターネットから実行オブジェクトをダウンロードして利用する、という局面で利用されることを想定している。

この種類に分類されるシステムとしては、ブラウザにダウンロードして実行される ActiveX Control や Java Applet などに対してデジタル署名を施すための Authenticode (Microsoft) が存在する。

2.2.3 コネクション型

前述のとおり、コネクション型とは、通信路のレベルで暗号化などの保護を提供する仕組である。

2.2.3.1 SSL

SSL (Secure Sockets Layer) [13]は、Netscape Communications 社が提案したセキュリティ・プロトコルである。現状では主としてウェブ・サーバとクライアント (ブラウザ) との間の通信を保護する仕組として利用されているが、実際のところは OSI 参照モデルで言うところのトランスポート層もしくはセッション層の辺りに相当するレイヤで動作するプロトコルであるため、ウェブで主に利用されている HTTP のみにとどまらず、たとえば TELNET や FTP、そしてメール (SMTP, POP3, IMAP) など、上位のプロトコルは自由に選択することが可能である。

プロトコルのバージョンとしては SSL2 と SSL3 とが存在するが、SSL2 は基本的に過去との互換性を保つためだけに用いられるべきものである。サーバ側の認証のみを行うものを SSL2、それに加えてクライアント側の認証をも行うものを SSL3 と呼ぶというような誤解がかなり一般的にあるように思われるが、実際のところは初期バージョンである SSL2 にはいくつかのセキュリティ・ホールが存在し、それを修正したものが SSL3 であると言えるため、今後 SSL2 は利用しないことが肝心である。

また、SSL3 をベースとしてさらに改良しインターネットでの標準プロトコルを定めようという動きとして TLS (Transport Layer Security)[14]の仕様策定が進んでおり、現在、多くのソフトウェアでその初期バージョンである TLS1(実質的には SSL3.1 と呼ぶことができるだろう) の利用が可能になっている。

SSL の特徴としては、基本的にはセッション開始時に、利用する暗号方式などのパラメータおよびアプリケーション・データの暗号化／完全性保証などに用いるための秘密情報をサーバ／クライアントで共有するための交渉 (これを SSL Handshake Protocol と呼称) を毎回行ない、実際のセッションで通信されるアプリケーション・データは交渉時に定められた秘密鍵により共通鍵暗号方式を用いて暗号化するという点が挙げられる。その SSL Handshake Protocol の概要 (ただし、SSL3 の場合) を図 2-11に示す。

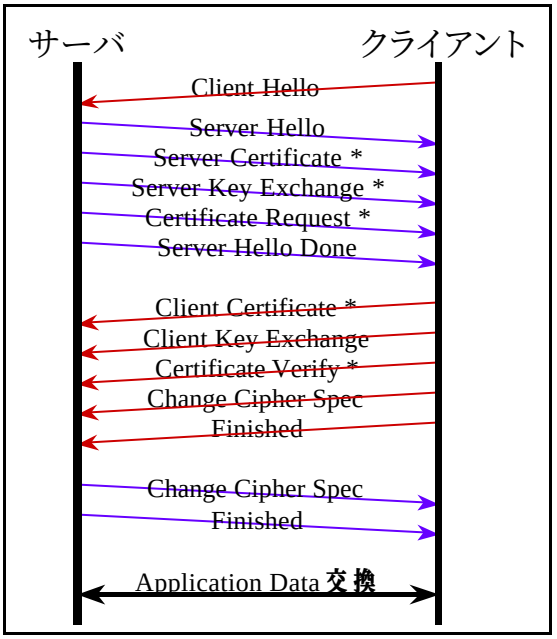


図 2-11 SSL Handshake 時のメッセージ交換

図で“＊”を記したメッセージはオプションであり、サーバ／クライアントの設定に応じて利用の有無が変更される。基本的には Client/Server Hello の交換で利用する暗号化方式などの交渉が完了し、Client Key Exchange メッセージによってその後生成される共通鍵などの基礎となるデータ (Premaster Secret) が、RSA や Diffie&Hellman 方式などの公開鍵暗号技術を用いて安全にクライアントからサーバへと送付される。Certificate Request、Client Certificate、Certificate Verify の３種類のメッセージは、オプションのクライアント認証を行うために必要となるメッセージである。

基本的にはセッションの開始毎に、以上のような交渉が行われることになるが、サーバおよびクライアントの設定によっては、以前のセッションで利用した各種パラメータをそのまま再利用するように図ることで、セッション開始時のオーバヘッドを低減することもできるようになっている。

2.2.3.2 IPSec

IPSec[15] は、OSI 参照モデルでいうところのネットワーク層にあたる IP のレイヤでデータの暗号化や完全性の保証、そして通信相手の認証を行うためのプロトコルである。

特徴として、相手の認証および暗号化で利用する鍵等の秘密情報の交換を、IPSec を用いる通信が発生するより前の時点で行なっておき、実際の通信時にはあらかじめ定まっているパラメータを用いてただちに暗号化などの保護が開始されるという点が挙げられる。これは、IP という低いレイヤでのセキュリティ機能提供となるためになるべく通常通信時のオーバヘッドを下げるためでもあるし、また、SSL とは異なり IP はコネクションレスの通信であるために SSL のようなネゴシエーションを行うタイミングが存在しないという事実もこの仕様決定に際して大きな影響を与えたのであろう。

このような構成となっているために、鍵交換手法としては、管理者による手動設定、

Kerberos、そして、公開鍵システム (Diffie & Hellman ベース) など、どのような方式でも採用できるようになっている。標準的な鍵交換方式として、IKE (Internet Key Exchange)[16] と呼ばれるプロトコルも提案されている。

実際の通信を保護するためには、AH (Authentication Header) と ESP (Encapsulating Security Payload) の二種類が IP で運ばれる新しいデータタイプとして定義されている。

2.2.3.3 無線電話

無線電話における暗号利用の例としては、現在 A5 と呼ばれる LFSR (Linear Feedback Shift Register) を利用したストリーム暗号アルゴリズムと、A8 と呼ばれる認証アルゴリズムが GSM (Global System for Mobile Communications) システムで用いられていることが挙げられる。

また、新しい規格として現在 IMT2000 の策定が進められているが、ここでは現行規格の問題点として

- subscriber authentication → アルゴリズム的に問題
- air-interface encryption → 鍵長が短すぎる
- subscriber identity confidentiality → より強化する必要性

というような点が挙げられている。現在、W-CDMA (Wide-band Code Division Multiple Access) をベースとする 3GPP (Third Generation Partnership Project) と、IS-95 CDMA, cdmaOne をベースとする 3GPP-2 が併存している状態であり、両者で標準化活動が進められている。

現状と同じく、共通鍵暗号ベースとなる見込みだが、3GPP-2 では認証部分に楕円曲線暗号が利用される可能性があるかもしれない。

2000 年 1 月には、三菱電機開発の共通鍵暗号アルゴリズム MISTY をベースとし、これを移動通信システム用にカスタマイズした KASUMI[4]というアルゴリズムが、3GPP における標準暗号として同グループ暗号検討ワーキンググループにより採用されたとの発表がなされた (3.1.3章参照)。これにより、同アルゴリズムは国際標準の地位を得た初の国産暗号技術となったとのことだ。

2.2.4 ストレージ型

データをハードディスクなどのストレージ・デバイスにファイルとして格納する時点で暗号化などの保護を行うことをストレージ型と呼んでいる。たとえば、コネクション型のセキュリティ・プロトコルを用いる場合、当然通信途上のデータの保護は行なえるわけだが、そのデータは目的のアプリケーションに届く前に復号されてしまう。そのようなデータをそのままの形でファイル等に格納してしまうと、計算機そのものに対して物理的、もしくはネットワークを介してアクセスできる人間にとっては容易に中の情報を知ることが可能となる。そのような場合に役に立つのがストレージ型のセキュリティ保護策である。

2.2.4.1 暗号化ファイルシステム

暗号化ファイルシステムは、ユーザやアプリケーションに対しては透過的に、ファイルシステムのレベルでデータの暗号化および復号をサポートするシステムである。他のデザイン的な代替案としては、PGP などの暗号化ソフトウェアを利用してユーザが自動もしくは半自動的に暗号化／復号を行うというものと、反対に物理的な機器もしくはデバイス・ドライバのレベルで暗号化／復号をサポートさせるというものとがあり得るが、誤操作の危険性、柔軟性などから考えると、ファイルシステムという中間レベルでこれらの機能をサポートするというのが最も有望なアプローチと考えられる。

このようなシステムの代表例としては、UNIX 上で動作する CFS (Cryptographic File System)[17] が挙げられる。これは、図 2-12のようにファイルシステムの上位／下位インタフェースの間に暗号化／復号エンジンをはさんでやることによって、write や read などのファイルに対するシステムコールのレベルで暗号化や復号処理が実現されるというものだ。

他に、デスクトップ上での透過的なファイル暗号化／復号を可能にする製品として、RSA セキュリティ社の SecurPC などを始めとする商用アプリケーションがセキュリティ関連ベンダから提供されている。また、前出の PGP を用いて同様の機能を実現する PGPdisk というツールも存在し、これはバージョン 6 以降の PGP 製品には標準で添付されるようになっている。

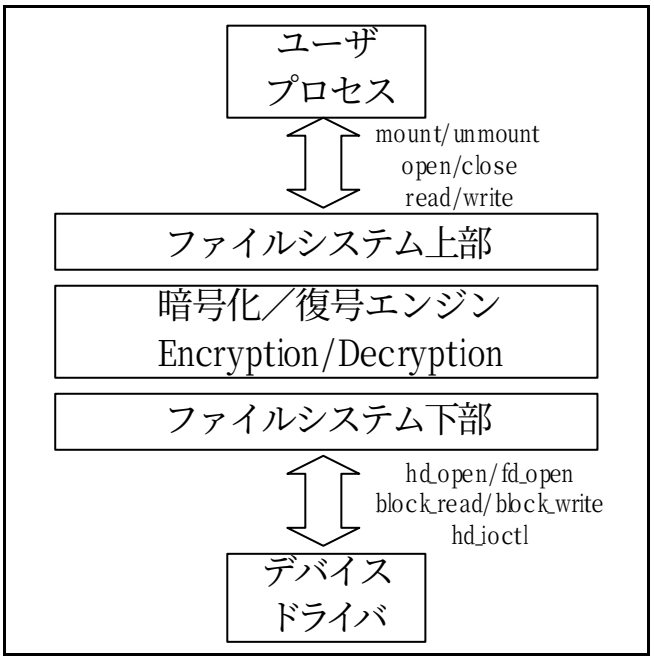


図 2-12 暗号化ファイルシステム

---

[4] = 霞 (MISTY の日本語訳)

2.2.4.2 ICカード

IC カード (欧米ではスマートカードという呼称が一般的) は、クレジットカード大のプラスチックカードに IC を搭載したデバイスである。これはさらに、純粋に記憶容量のみを提供するメモリカードと、専用の CPU を搭載しさまざまな機能を提供することが可能なインテリジェントカードとに分類できる。

メモリカードは純粋な外部記憶としてしか利用できないため暗号技術という面から見た場合は通常のハードディスクとなんら変わりがないので特に言及せず、ここではインテリジェントカードのみに話を絞る。

インテリジェントカードは独自の CPU を搭載しているため、応用的にも幅広く利用することができる。特に、暗号技術との関わりという観点から見た場合に重要なのは、鍵の保管デバイスとしての役割である。インテリジェントカードには中の CPU で鍵生成などの暗号化処理をサポートするものが存在するため、そのようなカードには、カードの中で鍵の生成から保存まで一切外部に情報を漏らすことなくすべての処理を実施することが可能なものがある。これらは物理的にもある程度の耐タンパ性を備えているため、ハードディスクやフロッピーディスクなどを鍵の保管場所として用いる場合と比べて強力な保護機能が提供できるため、次世代の鍵保管デバイスとして有望と考えられているのだ。そのため、後述の AES(Advanced Encryption Standard)選考では、アルゴリズムをスマートカード上に実装した場合の性能なども選定ポイントの一つとされている。

従来は個々のカードが独自の OS 機能を提供し、機能的にもさほど複雑なことができないものが多かったが、最近では MULTOS, Java Card、そして Windows for Smart Cards など汎用的な OS を搭載し、多機能を提供するカードが提供されるようになっているため、今後ますますの普及が予想されている。

ただし、IC カードの耐タンパ性に関しては、現状では十分系統立てた理論が確立しているわけではなく、IC 中に各要素をどのように配置し、外部から加えられる走査を無効化するための仕組を如何に巧妙に組み込むか、という点に関して、一種職人芸的な対策が練られているといった段階であって、将来に渡って安全性が確保できるかどうかは疑問視されているところでもある。

たとえば、1999 年３月に開催された 2nd AES Conference (AES 策定の一環として開催された会議) では、IC カード上にナイーブに実装された AES 有力候補の一つである Twofish が、Differential Power Analysis[18]と呼ばれる IC カードの消費電力を分析するというタイプの攻撃方法を用いることで、非常に簡単に破られたという報告がなされている。もちろん、これは Twofish というアルゴリズム自体に欠陥があった故ではなく、単にその IC カードへの実装方式が十分練られたものではなかったということが原因であり、報告は他の AES 候補にも同様の危険性があるとしているが、このエピソードなどは IC カードという比較的新しいデバイスに対しては、まだまだ多くの攻撃方法を開発する余地が残されているということを如実に示す実例と言えるのではないか。

3 暗号の仕組みと要素技術

3.1 共通鍵暗号（対称暗号・慣用暗号）

3.1.1 ブロック暗号方式

(1) ブロック暗号の定義と原理

ブロック暗号とは、平文を一定のブロック長毎に区切って、共通鍵を用いて暗号化処理を行い、復号の際も同じブロック単位に共通鍵を用いて復号を行うものをいう。平文がブロック長に満たない場合はパディングが必要である。パディング方法についてはアルゴリズム毎に定められた方法を用いることが、暗号の強度を保つためにも必要であるが、特に定められていない場合は用途に応じ決定する必要がある。

ブロック暗号は、暗号化の際にデータを攪拌するデータランダム化部と、共通鍵から各段のデータランダム化部へ供給するサブ鍵を生成する鍵スケジュール部から成る。データランダム化部は、解読を困難にするために様々な手法が用いられるが、現在多くのブロック暗号アルゴリズムにおいて、Feistel の開発した暗号で用いられたインボリューション(involution)と呼ばれる１：１変換のデータランダム化処理テクニックが用いられている。インボリューションの例を図 3.1-1 に示す。これらは 2 回繰り返したり、逆操作を行うことで逆変換が可能であるが、その操作内容を決定するのに暗号化鍵（または鍵から導出される複数のサブ鍵）を用いる。

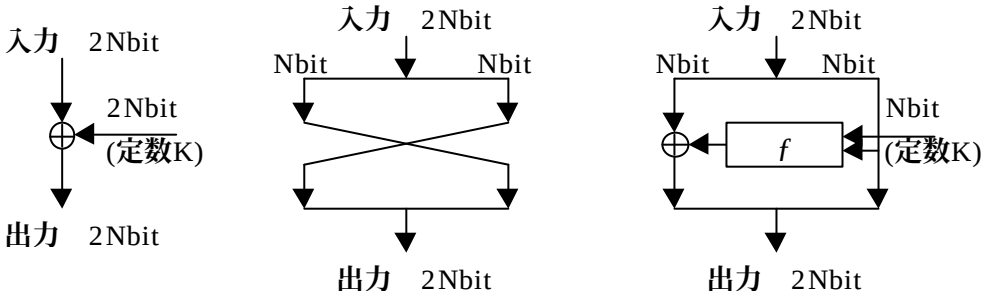


図 3.1-1 インボリューションの例

図 3.1-1(左)の⊕は排他的論理和(exclusive-OR ； XOR)を示しており、2 を法とする加算として、bit 毎に次の式によって定義される。

$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0$

2Nbit 長の bit 列 p と 2Nbit 長の bit 列定数 K の排他的論理和による出力を c とすると、排他的論理和の定義から明らかに

$p \oplus K = c$

$c \oplus K = p$

図 3.1-1(中)は、入力ブロック(2Nbit 長)の上位 Nbit と下位 Nbit を入れ替える操作である。

図 3.1.-1(右)の関数 f では bit を攪拌するための操作として、bit の置き換えや並び替え（位置の変更）が行われる。これら非線形な処理を繰り返し行うことで解読しにくい暗号文を生成する。

一般的な Feistel 型暗号の構成を図 3.1-2 に示す。

データランダム化部では、平文 P のブロックを上位部 Li と下位部 Ri に分割し、インボリューションから構成される各段の変換を反復する。後述の鍵スケジュール部では、鍵 K(マスター鍵)をランダム化し、データランダム化部の各段へサブ鍵 Ki を供給する。

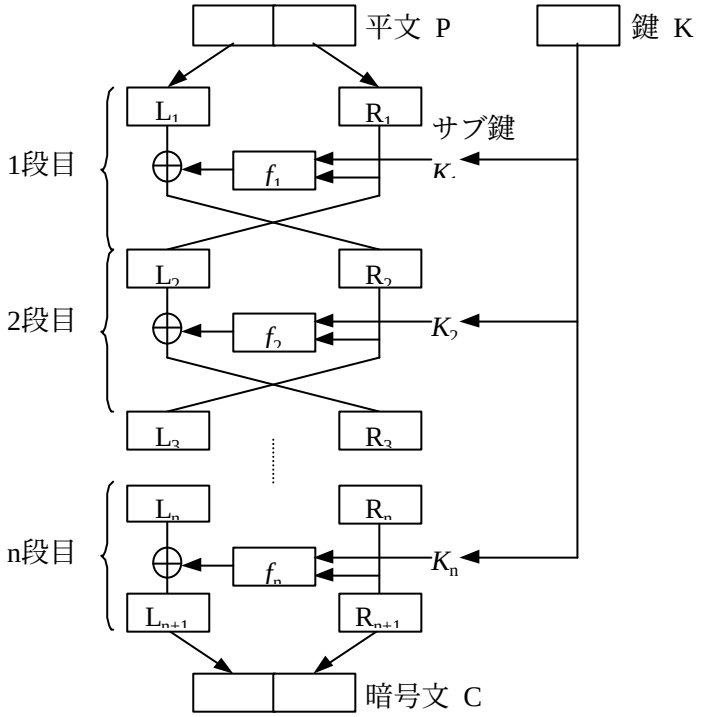


図 3.1-2　Feistel 型暗号の基本構造

ここで暗号アルゴリズムの強度を決定する要因としては鍵長のほかに、段数、各段の *f* 関数、サブ鍵がある事が分かる。

DES などでは S-box と呼ばれるテーブルを参照し、その内容にしたがって bit の並び替えを行う。この方法はハードウェアで実装するには都合が良いが、ソフトウェアで処理するには非常に時間がかかる。そのため、最近の主要な暗号方式では CPU 処理の軽い bit シフト操作（Rotation）を用いるものも多い。

一般に、出力暗号文の各 bit は平文と鍵のより多くの bit の影響を受けるよう設計する。そのために、図 3.1-2 のように複数段繰り返して行うことが必要であるが、段数（ラウンド数とも呼ばれる）が増加すると処理時間が増大するため、少ない段数で攪拌効果を高めるために各アルゴリズム毎に工夫されている。処理ブロック長はアルゴリズムにより固定のものと可変のものがあるが、可変の場合も、処理効率からインプリメント毎に固定とすることが多い。また、最近のアルゴリズムでは段数も可変とし、暗号化対象の重要度にあわせて暗号強度を変えられるにできるようにしたものがあるが、用意する S-box やサブ鍵、処理テーブルの数が影響を受けるため、固定としたものも多い。一般に、ブロック長が長くなると攪拌効果が低下するため、段数も増加しなければならないと考えられている。ブロック長決定においては安全性を考慮し、各アル

ゴリズムで推奨される段数以上を用いた方がよい。

暗号化鍵（マスター鍵とも呼ぶ）から各段で用いるサブ鍵を生成する処理は、鍵スケジュール処理（Key Scheduling）と呼ばれる。鍵スケジュール処理では、元の暗号化鍵からサブ鍵を求めるのに以下のようないつくかの手法がある。

- マスター鍵をｎ分割しｎ個のサブ鍵とする（サブ鍵の総 bit 数はマスター鍵と同一）
- マスター鍵（またはその分割したブロック）を種として bit 拡大のための一方向性関数（3.3　ハッシュ関数）を用いる（サブ鍵の総 bit 数はマスター鍵より多い）

近年、この鍵スケジュール処理アルゴリズムにより、類似した暗号化鍵を使用する場合の安全性についての研究が行われるようになった。その結果鍵の差分攻撃ともいえる方法で鍵を推定できることが明らかになった[1]、[2]。

この中で、安全な鍵スケジュール処理の条件として以下のものが挙げられている。

- サブ鍵を求める処理が線形変換でないこと（逆演算不可能なこと）
- マスター鍵の各 bit がサブ鍵に対して同じ影響力を持つこと
- 生成されるサブ鍵間に依存関係が無いこと

(2)　ブロック暗号方式の利点と欠点

共通鍵暗号自体の得失については 2.1.1.1(1)②に述べたとおりで、ここではブロック暗号での得失について述べる。

A．利点

ブロック暗号は、もともとハードウェア化する事を前提に開発された経緯があり、そのため固定のブロック単位で処理を行うようになっている。すなわち、ハードウェア化が容易で、チップも比較的入手しやすい。ソフトウェアで実装する場合も入出力のブロックが一定であり、開発・検証が容易でありコードも比較的単純である。

また、同じブロック長と鍵長のアルゴリズム同士の場合、簡単に別のアルゴリズムに切り替えることができるメリットがある。さらに、最新の暗号方式では、他の方法に比べて使用するメモリなどの容量が小さくてすむように工夫されているという特長がある（ストリーム暗号では鍵の展開や乱数生成のために多くのエリアが必要となるし、公開鍵暗号方式は計算量が多いためコードも大きくなるだけでなく、鍵やデータのバッファサイズが極端に大きくなってしまう）。

B．欠　点

先に述べたとおりブロック暗号方式では同一の鍵と平文の組み合わせでは必ず同一の結果が得られる。特に、ブロック長よりも長い平文を暗号化する場合、3.1.6 に述べるブロック暗号の利用モードを選択する際に、ECB モードと呼ばれる基本的な利用モードを使うと、冗長データ（Null、スペース、０など）が多い場合に同じ暗号文ブロックが発生し、平文内容が推定される恐れがあり、解読が容易になる可能性がある。また、ブロック単位で処理を行うため、暗号文中ではブロック境界で偽の情報を埋め込まれたり情報の一部を消されて改竄されても、受信者に判別できないという欠点もある。

ECB モード以外の利用モードでは、こうした欠点を克服するために初期化ベク

タ(IV)やブロック連鎖を用いる。この中で CBC モードは比較的ポピュラーであり、多くの暗号アルゴリズムで使用することができる。また、RC5 など一部の暗号方式では CBC 同様の連鎖(チェーニング)処理を組み込んだものもある。またストリーム暗号と異なり、ブロック長に合わせるためのパディングが必要となるが、その処理が不適切だと平文の内容に偏りが生ずる恐れがある（上記冗長データと同じ）。

ブロック暗号アルゴリズム全体の弱点としては、鍵の総当たり法以外に次のような既知の攻撃法(ショートカット)があり、各アルゴリズムに対してはこれらの攻撃法による強度低下の研究や、対抗するために bit 攪拌方式を工夫している。AES 候補など、最近のブロック暗号アルゴリズムでは、線形特性確率や差分特性確率を算出し、これら既知の攻撃法に対する安全性を理論的に証明することが一般的となってきた。

C. 線形解読法

1993 年に三菱電機の松井によって報告された解読法で、既知平文攻撃（暗号文と平文のペアから推定する方法）の一種。平文と暗号文の幾つかの bit の排他的論理和と鍵の特定 bit との相関関係より転置・換字のパターンを推測する。

D. 差分解読法

1990 年にワイツマン研究所の Biham と Shamir が報告した選択平文攻撃（任意の平文に対する暗号文を入手できる場合の方法）の一種。都合の良い複数の入力の差分（異なる bit）を選び出し、それに対する暗号文の差分の統計的性質より転置・換字のパターンを推測する。のちに、DES の開発時には考慮されていたことが判明した。

上記解読法はいずれも数百億個の平文と暗号文のペアを入手しなければならず、それだけのデータ量を収集し保管・検索することは、現実的な時間内では困難である。上記報告内容も論理的な考察であり、実際に DES が（公の場で）解読されたことは、つい最近までなかった。

また、通常はこれだけ多くのデータの暗号化を一つの鍵で行うことは、万一鍵が解読された場合に非常に多くのデータが解読されることを意味するため大変危険であることは良く知られており、通常は一定期間（または回数）で鍵の変更が行われる。

一方、総当たり法での解読には膨大な回数の試行が必要であるが、たった一つの暗号文と平文の組だけがあれば可能である（実際には平文は正確に内容が分かっている必要はなく、平文の推測される範囲、たとえば全てアルファベットである、全て数字である、あるいは記号を含む英数字であるといった、機械的に判定できるものに限定できればよい）。

この事からも、平文の内容が推定されるようなデータ（連続文字または bit パターン列）にブロック暗号を施す場合には注意が必要である。後述する実際の DES の解読でも、暗号文のみが与えられた状態で解読作業を行い、意味のある英文を取り出すことに成功したため、解読できたと判断することができた。逆に言えば、平文の内容がまったくランダムな bit 列であった場合は、それが正しく解読できたかどうかを他の手段で確認できなければ、攻撃者は解読に成功したこともわからないといえる。

3.1.2 各種ブロック暗号の比較

ここでは、現在主に用いられている、あるいは今後有力になると見られる、各種ブロック暗号について簡単に内容をまとめる。

表3.1-1　主な共通鍵ブロック暗号方式の比較

ISO9979 登録番号	名　称	キー長	ブロック長	段数	開発元（パテント）*1	発表年	速　度*2
0004	DES	56bit	64bit	16	IBM（特許放棄 *3）	1977	7.7Mbps(P5-90) *4
	TripleDES（２キー）	56bit×2 (57bit 相当)	64bit	16×3	同上（特許なし *5）	1977	DES の 1/3
	TripleDES（３キー）	56bit×3 (112bit 相当)	64bit	16×3	同上	1977	DES の 1/3 2.6Mbps(P5-90) *4
0010	FEAL-*N*	64bit	64bit	N	ＮＴＴ	1987	7Mbps(P5-90)*6 55Mbps(専用 LSI)
	GOST 28147-89	256bit +512bit	64bit	32	Zabotin,Glazkov, Isaeva	1989	
0009	MULTI2	256bit	64bit	可変	日立製作所	1989	
0002	IDEA	128bit	64bit	8	Ascom Systech	1991	DES の 2 倍 177Mbps(専用 LSI)
0012	SXAL8	64bit	64bit	8	ローレルインテリジェントシステムズ	1991	
0013	MISTY1 MISTY2	128bit	64bit	8 *7	三菱電機	1992	20Mbps(P5-100)
0012	MBAL	64bit	16-1,024 Byte	3	ローレルインテリジェントシステムズ	1993	24Mbps(P5-133)
RFC2040	RC5	1-256Byte	32,64,128 bit	可変	RSA Security	1994	24Mbps(P5-90)
0006	Skipjack	80bit	64bit	32	ＮＳＡ	(1994) 1998*8	
0008 RFC2268	RC2	1-128Byte	64bit	18	RSA Security	(1994) 1998*8	8Mbps(P5-90)
	SAFER	64bit	64bit	6~	Jim Massey（フリー）	1994	
	Blowfish	32-448bit	64bit	可変	Bruce Schneier（フリー）	1994	25Mbps(P5-150)
RFC2144	CAST	40-128bit	64bit	可変	Entrust Technologies（フリー）	1997	26Mbps(P5-150) CAST-128
0019	CIPHERUNICORN-E	128bit	64bit	16	ＮＥＣ	1998	5.7Mbps(PentiumPro-200)
0014	ENCRiP	64bit	64bit	可変	ＮＥＣ	非公開	
	Khufu / Khafre	64bit	64bit	8×ｎ	Xerox	非公開	
0017	SPEAM1	128bit	64bit	4×ｎ８～	松下電器産業	非公開	
0018	ELCURVE	128bit			日立製作所	非公開	
0020	M8	64bit～	64bit	可変	日立製作所	非公開	
AES候補	MARS	128-1,248bit	128bit	32	ＩＢＭ	1998	65Mbps(PentiumPro-200)
AES候補	RC6	2,040bit 以下	可変	可変(20)	RSA Security	1998	100.8Mbps(Pentium Pro-180)
AES候補	Rijndael	128,192,256bit	128,192,256 bit	10,12,14	J.Daemen and V.Rijmen(フリー)	1998	80Mbps(Pentium-200 Assembler)
AES候補	Serpent	256bit 以下	128bit	32	R.Anderson, Eli Biham, L.Knudsen(フリー)	1998	15Mbps(PentiumPro-200 Borland C++5.0)
AES候補	Twofish	128,192,256bit	128bit	16	B.Schneier, J.Kelsey, D.Wagner, N.Ferguson ら(フリー)	1998	90Mbps(PentiumPro-200 Assembler)

*1　開発元は主に知的所有権保有者を記す。
*2　速度は注記以外開発元による公表速度
*3　DES 自体の特許については放棄しているが関連特許があるので注意
*4　RSA 社製 BSAFE ツールキットによる値
*5　月刊インタフェース 1997 年 9 月号 pp.118
*6　FEAL 32 の値
*7　実際には入れ子構造のため８(32bit)×３(16bit)×３(7,9bit)＝72 段の S-box を用いる
*8　アルゴリズムが公開された年

3.1.3 各ブロック暗号の概要

(1) DES

DES（Data Encryption Standard）は、現在世界中で最も広く使われている暗号方式である。後述のように、1997 年６月に RSA 社の暗号解読コンテストにおいてついに解読されるということになったが、解読作業そのものは総当たり法による解読であり 5.2.3 で述べる計算量的安全性評価の範囲内であったため、鍵長の問題は別にして DES の暗号化方式に対する脅威とは見なせないであろう。

元々は IBMM社が 1970 年代前半に開発し、1977 年に行われた NBS（米国商務省標準化局：National Bureau of Standards、現在の NIST）の暗号アルゴリズム公募に提案されたものを修正したものである。当初は専用ハードウェアを必要としその内容も完全には公開されていなかったため、政府による解読のために、秘密の抜け道があるのではないかと疑われていた。その後、仕様が完全に公開され、プロセッサの能力向上と合わせてソフトウェアによる実装も多く見られるようになってきた。ただし、仕様が公開された後も、DES の中で用いられる換字処理のテーブル設計に疑問点があることが指摘され、この点も政府による解読のためではないかという議論も行われた[3]。現在では、DES の設計方針が差分解析などの解析技法を研究の上決定されたことが明らかになったことから、そのような疑いはなくなったようである。

DES 自体の特許は標準規格採用時に IBM 社が放棄し、また関連特許もほとんどが期限切れであるため現在は自由に使用できる安全性の保証された暗号方式という位置付けになっている。

DES の仕様そのものは先に述べた Feistel の開発した暗号の応用である。暗号化／復号処理は 16 段の換字（f 関数）と 32bit 転置処理からなる。ただし、DES の特長としては各段に用いるサブ鍵を、56bit 鍵→パリティを付加した 64bit 鍵→分割した 32bit サブ鍵→48bit サブ鍵へ拡大転置という手法で作成し、この 48bit サブ鍵で bit の置き換え処理を行っていることである。このように、複雑な処理を行うためソフトウェアによる実装では十分な速度が得られ無いことが問題であったが、多くの企業・学者の努力で各処理の高速化とテーブル処理の追加により現在では多くの高速化ルーチンが入手可能である。前記のデータも RSA 社が自社の BSAFE ツールキットでアセンブラによる最適化処理を行った結果であり、高級言語で仕様書どおりに組んだ場合に比べて５倍程度高速といわれている。

なお、DES は途中で一部仕様が改訂されており、現在でも文献により S-box の設計が異なるものがあるため注意が必要である。これらは、それ自身暗号化と復号が正常に行われたように見えるが暗号文が異なるため注意が必要である。

実装および検証にあたっては、以下の公開標準ドキュメントの内容に照らし合わせて確認することをお勧めする。

- FIPS PUB 46-3 DATA ENCRYPTION STANDARD (ＤＥＳ) [4]
- FIPS PUB 74 - GUIDELINES FOR IMPLEMENTING AND USING THE NBS DATA ENCRYPTION STANDARD [5]
- FIPS PUB 81 - DES MODES OF OPERATION [6]

(2) IDEA

IDEA（International Data Encryption Algorithm）は、インターネット上でチューリッヒにあるスイス連邦技術研究所の Lai と Massey が Ascom 社と共同開発した暗号アルゴリズムで 1991 年に発表された。現在権利は Ascom Systec 社が保有している。IDEA はもっとも普及している暗号メール PGP の標準アルゴリズムとして採用され知名度が高くなった。IDEA では 128bit 鍵を用いるため総当たり攻撃でも今後数十年安全を保てる可能性があることが特長である。また、内部処理では 64bit 平文を４つの 16bit に分割し、加算・乗算・XOR・剰余処理を行っており、マイクロプロセッサとの親和性を重視していることがわかる。専用チップでも 177Mbps という高速性を発揮し DES に対する優位性が強調されている[7]。また、128bit の鍵から 52 個の 16bit サブ鍵を生成し、各段で６個のサブ鍵を用いて８段の処理を行い、最後に４つのサブ鍵で最終変換を行うという構成になっている。

IDEA は日米欧で特許が成立しており商用利用の場合はライセンスが必要であるが、PGP のようなフリーソフトで使用する場合は申請のみで使用料が不要である。

(3) FEAL-*N* / *NX*

FEAL（Fast Data Encipherment Algorithm）は 1987 年に NTT で開発された暗号方式で、８bit マイクロプロセッサ上で高速に処理できることが特長である。FEAL には 64bit 鍵を用いる FEAL-*N* と 128bit 鍵を用いる FEAL-*NX* がある（ブロック長はいずれも 64bit）が、FEAL-*N* は FEAL-*NX* の鍵の下位 64bit を０としたものである。*N* は段数で４以上の偶数であるが８、16，32 が一般に知られている。

内部処理では、128bit 鍵から N+８個の 16bit サブ鍵を求め、初期処理として４個、各段１個、最終変換で４個のサブ鍵が使用される。各段では *f* 関数内で鍵および入力データをそれぞれ２個と４個の８bit ブロックに分割し、XOR、加算、剰余、２bit 左巡回シフト操作を行い、結果の８bit×４ブロックを 32bit 出力としている。

FEAL８および 16 に関しては差分解読法および線形解読法での解法が示されているが、FEAL32 に関してはいずれも不可能であることが証明されている。FEAL は NTT が開発したこともあり、国内で安全に使用できる強力な暗号としての選択肢に入るものである。また、専用チップも国内市販されており応用製品も販売されているが、８bit 処理を多用しているため上記 IDEA チップに比べて１／３の能力にとどまっている。ただし、これまでに多くの解読実績があることから、FEAL の安全性を疑問視する向きも多く、今後の信頼性回復が最重要課題である。

(4) MULTI２

MULTI２（Multimedia Encription２）は日立製作所が 1989 年に発表した暗号方式である。256bit という非常に長い鍵を使用することが特長である。

基本構造として 32bit プロセッサ上での高速処理を目指したもので、同様のアルゴリズムがデジタル衛星放送などで使用されている。各段では、32bit の加減算・XOR・OR・右巡回シフト（１，２，８，16bit）を行って、攪拌効果を高めるよう工夫されている。また、利用者が安全性と処理速度のバランスで繰返し回数を選択できるようになっている。

なお、日立製作所では MULTI３以降の暗号については、受託開発方式であるとしておりアルゴリズムも公開されていない。

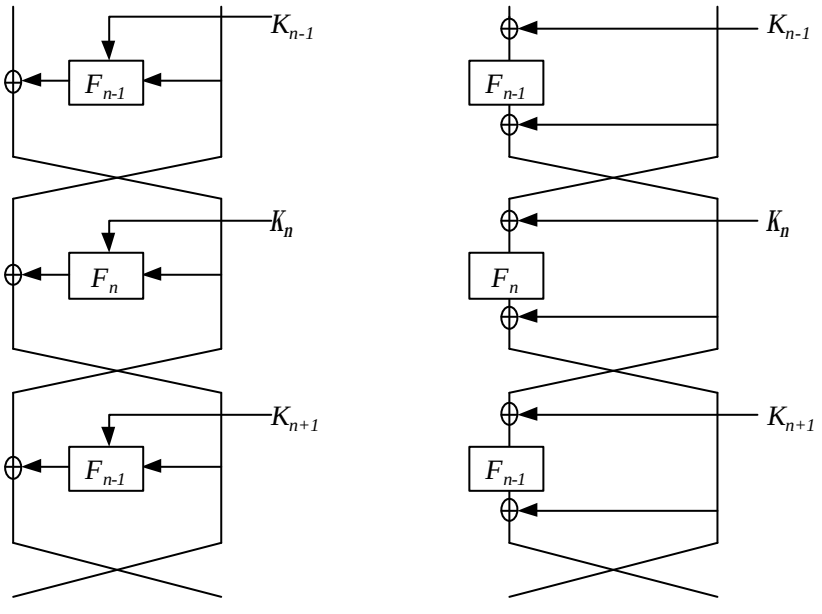
(5) MISTY

MISTY は三菱電機が 1992 年に発表した暗号アルゴリズムで、MISTY の名前は開発者の頭文字をつなげたものである。MISTY では 128bit 鍵から 256bit の拡大鍵を生成して使用し、ガロア体上の演算を基本に演算要素として AND，OR，XOR とテーブル参照のみで F 関数を設計している。差分解析法および線形解析法（開発者の一人・松井が考案）に対する証明可能安全性を持つことが特長である。

これは、MISTY が３重の入れ子構造で、サイズの小さい、平均差分／平均線形確率の小さなテーブルを組み合わせて構成しているために可能となった。MISTY には MISTY１と MISTY２の２種類あり、MISTY１は通常の Feistel 型であるが、MISTY２では処理の並列化が可能になるよう、各段図 3.1-3 のように並べ替えて実装している点も特長である。

ここで各段のＦ関数（32bit 入出力：MISTY では FO 関数と呼ぶ）は、さらに３段の 16bit 入出力の F1 関数からなり、FI 関数はまた３段のＳ関数（S7 および S9）からなるという構成をとっている(図 3.1-4)[8]、[10]、[11]、[12]。

第 3 世代移動通信システム(W-CDMA)の国際規格を検討している標準化プロジェクト 3GPP(3rd generation partnership project)の暗号検討 WG は 2000 年 1 月、MISTY を移動通信システム用にカスタマイズした KASUMI を 3GPP の標準暗号として採用した。



MISTY1の基本構成　　MISTY2の基本構成

図 3.1-3　MISTY の基本構成

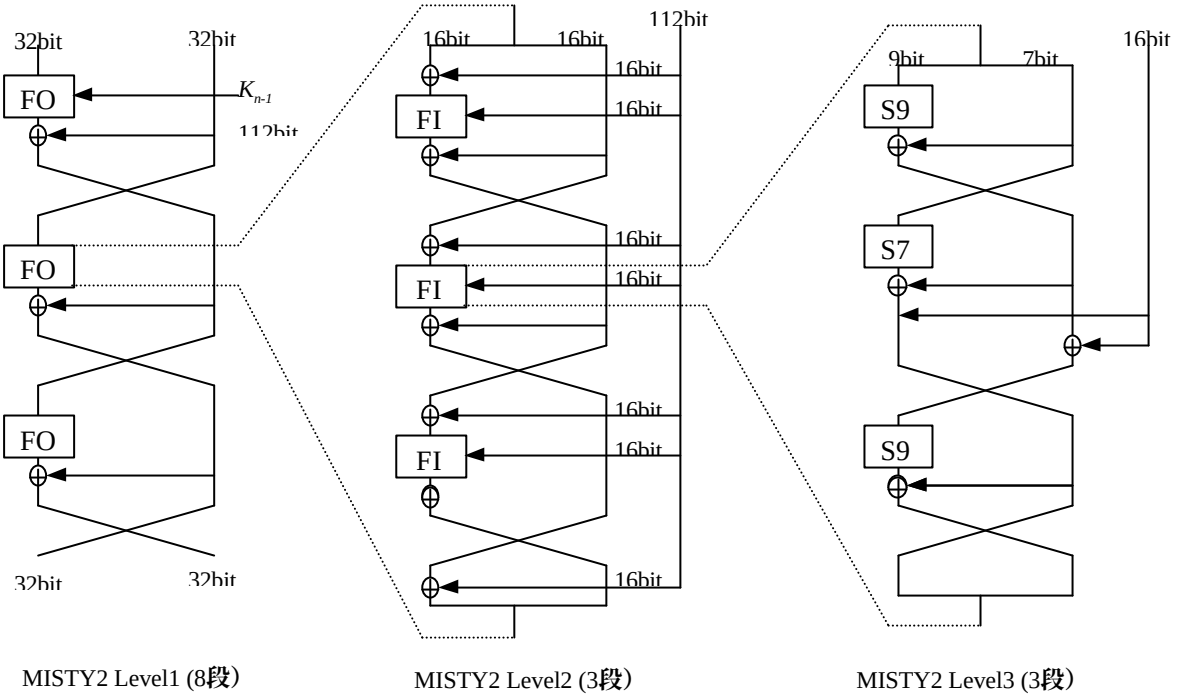


図3.1-4　MISTY の入れ子構造

(6)　SXAL／MBAL

SXAL(Substitution Xor Algorithm)と MBAL(Multi Block algorithm)はともにローレルインテリジェントシステムズ社の平田氏の考案になるもので、内部処理は 8bit 単位の換字・XOR・転置処理で構成されるため、マイクロプロセッサ上のソフトウェアで高速に実現できる暗号方式である[13]。

SXAL(段数を表し SXAL８とも呼ばれる)では、サブ鍵として初期処理と最終変換に 2 個、各段に１個使い合計 10 個のサブ鍵が用いられる。各段は図 3.1-5 のように 32bit ×2 入力 2 出力の基本アルゴリズムｆで、入力の左右両方を換字・XOR する処理である。一方ＭＢＡＬではこの基本アルゴリズムを拡張して可変長 Byte の入力に対して処理を行う拡張基本アルゴリズム $F_m(K)$が用いられ、更に段数を３回に減らしても安全性を確保できるよう、段間で両端データを SXAL で暗号化処理を行う[9]。

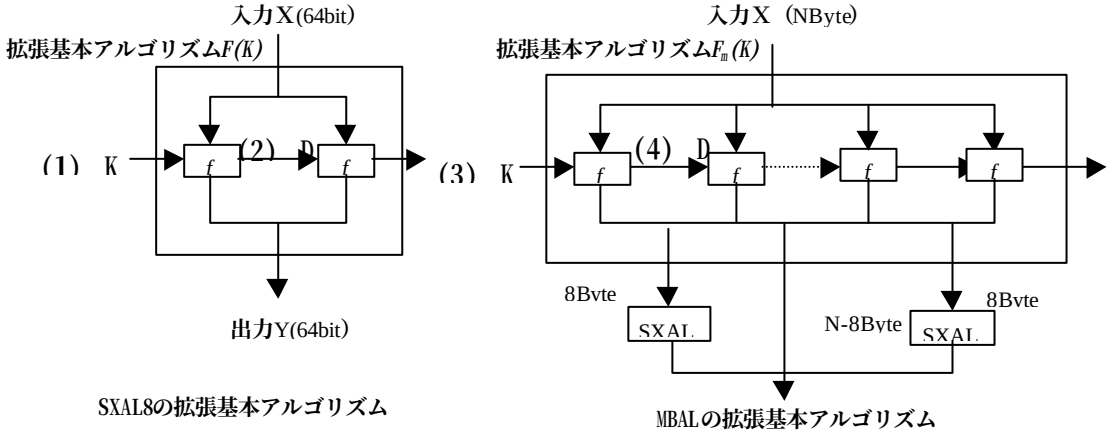


図3.1-5　SXAL/MBAL の基本アルゴリズム

(7) RC2

RCx という暗号方式はＲＳＡ開発者の一人である Rivest が DES の欠点を克服するために開発した共通鍵暗号方式である。現在 RC2, RC5（ともにブロック暗号）、RC4（ストリーム暗号）の３つが広く使われている。また、RC1 は実現できず、RC3 は開発中に破られたため欠番となっている。RC4 は AES 最終候補に残っている。

RC2 は長くそのアルゴリズムが非公開とされ、RSA 社のツールキットを使用しなければならなかった。ただし、鍵長が可変であるため、米国からの輸出規制(当時 40bit)に早くから適合し、製品に採用されている場合も多い。国内でも DES が解禁される前からポピュラーな存在であった。

最近になって IETF での標準化の提案にあわせて、RC2 のアルゴリズムが Rivest 自身により公開された[14]。これによると、RC2 は１～128Byte の可変長の鍵を使用可能で、64bit ブロック単位に処理を行う暗号方式である。また、内部の処理単位として 16bit 語のマイクロプロセッサ上で実装が簡単になるように考慮されている。鍵の展開処理においては 16bit 長、８bit 長どちらでも処理できるようになっているが、ランダムなバイト列を作成するために、π（円周率）に基づく 256Byte の数列を用いている点がユニークである。

RC2 の攪拌処理は単純な繰返しではなく、５回の Mix 処理－１回の Mash 処理－６回の Mix 処理－１回の Mash 処理－５回の Mix 処理という上下対称の処理ロジックとなっている。Mix 処理では、加算・AND・左巡回シフト操作（１，２，３，５bit）を行い、Mash 処理では加算とＡＮＤ処理を行う。

(8) RC5

RC5 は Rivest の開発した暗号方式で、処理速度、スケーラビリティに配慮した処理を基本とし、ブロック長、鍵長、段数がいずれも可変という特長を持つ。これらは、処理ブロックプロセッサのワード長の２倍で処理することで高速性を持たせ、鍵長と段数は処理速度と安全性のバランスで利用者が決められるようにしたものであり、ブロック長により十分な攪拌効果を得るための段数が変わる。RC5 の基本アルゴリズムは驚くほど簡単で、XOR 操作、データ依存左巡回シフト操作、サブ鍵の加算、左右交換処理だけからなっている。このため、コードサイズ、メモリサイズいずれも少なくすることができ、IC カードなどでも利用することができる。

RC5 の基本的な安全性はデータ依存のシフト操作によるものであるが、未だ発表から年月が経っていないこともあり安全性についての研究はまだ十分であるとは言い切れない。また、実装例では CBC 処理を基本処理に組み込んでおり、データ依存シフト処理の危険性回避と見ることもできる。最近になって、ある条件を満たした場合の差分解読の可能性[16]や、入力平文の一部を固定した場合に生じる出力暗号文の偏りを c^2 (カイ二乗)検定により検出し、選択平文攻撃と併用することにより最終段拡大鍵が推定可能であること[17]などが示されている。

また、RC5 は当初よりアルゴリズムが公開されているが、RSA 社は商標および特許(US5724428: Block encryption algorithm with data-dependent rotations)を登録している。ただし、RSA 社の特許に関する権利の放棄を IETF が標準化採用の見返りと

して要求していることもあり、今度の動向が注目される。なお、日本国内においては RC シリーズ暗号は出願されていないようである。RC5 の完全な仕様書とプログラム例が IETF により[15]に公開されている。

(9) ENCRiP

ENCRiP は NEC が開発し 1997 年に ISO 登録を行った 64bit 鍵、64bit ブロック長の暗号方式である（段数については登録内容は８段であるが詳細は不明）。

詳細が公開されていないため、サブ鍵の生成などの処理内容は不明であるが、基本となる攪拌処理内容について関連特許（特開平 9-054547）が公開されている。それによれば、内部の基本処理としては 32bit 単位での XOR，AND、左巡回シフト操作（１，７，11，22bit）で構成されており、特許出願意匠もこの組み合わせによる高度な bit 撹乱効果である。シフト操作が１，７（=８－１）、11（=32+１÷３）、22（=11*２）と工夫することで、１段あたり非常に多くの入力と鍵の bit が出力 bit に影響を及ぼす設計となっているようである。

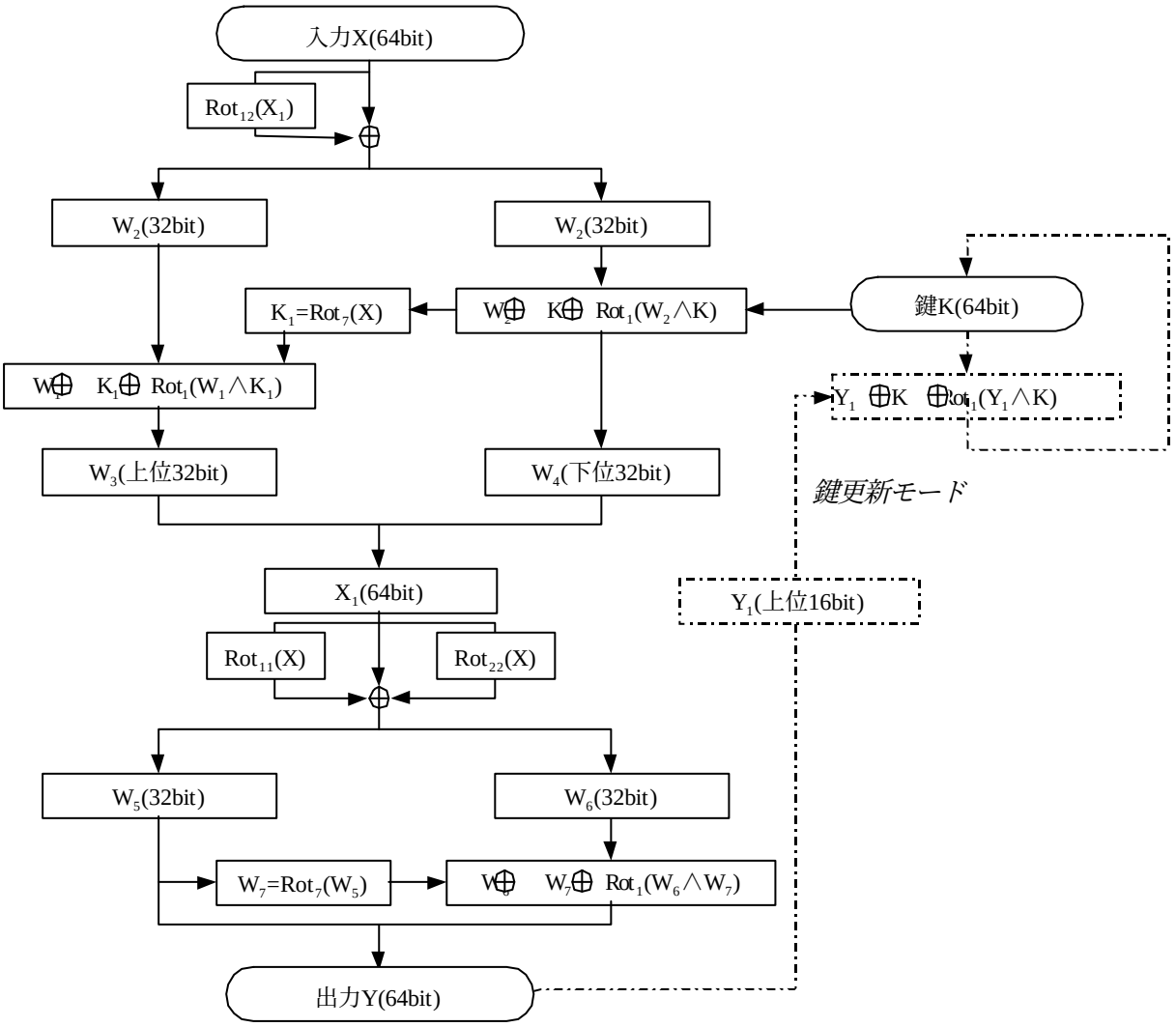


図3.1-6　ENCRiP で使用されるアルゴリズム

(10) SAFER

SAFER（Secure And Fast Encryption Routine）は、IDEA の開発者の一人 Massey が Cylink 社のためのオープンな暗号方式として開発したものであり、現在はフリーで使用することができる。また、鍵長を表し SAFER K-64 と呼ばれることもある[18]。

SAFER の特長は、全て８bit 単位で処理していることと、擬似 Hadamard 変換（PHI：Pseudo-Hadamard Transform）と呼ばれるあまり知られていない線形変換処理をブロック全体にたすきがけしている点、そして 45 を底とする指数・対数演算と 257 を基数とする剰余処理を採用している点である。実際にはこれらの処理はテーブル参照により行われるため処理そのものは高速である。

段数は可変となっており６段以上の任意の値にすることができる。各段では 64bit サブ鍵を２個、最終変換で１個使用し、１つの 64bit 鍵から $2r+1$（r は段数）個のサブ鍵を生成して使用する。サブ鍵の生成では鍵バイアスというテーブル参照で弱い鍵をなくすようにしている。

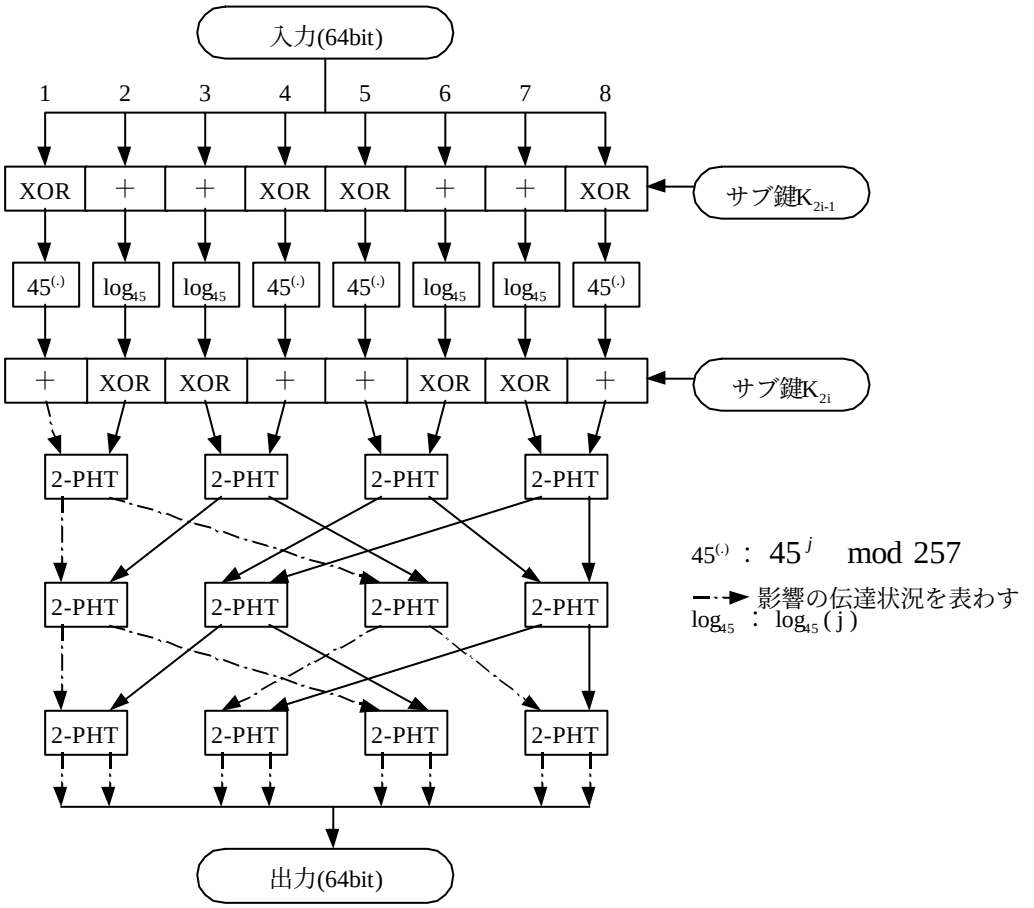


図 3.1-7　SAFER の各段の処理

(11) Blowfish

Blowfish は民間の暗号学者で”Applied Cryptography”の著者として知られている Bruce Schneier が開発しフリーソフトとして公開している 64bit ブロック暗号方式で

ある[19]。

鍵長は可変となっており 32bit から 448bit まで可能である。また、段数は 16 段固定で、各段で１個、最終変換で２個の 32bit サブ鍵を使用する。S-box は 32bit のものを４×256 個使用する。特長は高速性であり、１Byte あたり処理クロック数は RC5 や Khufu よりも少ないといわれている。*f* 関数として８bit ブロックでの加算、XOR、232 を基数とする剰余を利用している。

最近 IETF にも標準化採用のための提案がなされており、ブロック長を伸ばす改良が加えられているとも伝えられているが、安全性に対する評価は定まってはいない。

(12) Skipjack

Skipjack は、NSA が Capstone プロジェクトで使用する Clipper チップで使用するために開発した暗号化アルゴリズムである。単純に安全性を比較すると 80bit 長のキーを使用して 32 回データを攪拌するため DES よりも安全性が高いと考えられる。

当初 Skipjack のアルゴリズムについては公表しない方針が採られたため、ソフトウェアでの実現は不可能で、認可されたチップメーカの製造するハードウェアでのみ実現可能なため、採用には非常に大きな問題点となった。特に、アルゴリズムが公開されない点は、政府のみが知る解読法を隠すためではないかという批判を受け、政府は、民間の暗号学者に Skipjack のアルゴリズムの研究を依頼し、レポートが公開された[20]。これによれば「開示された範囲で調査した結果、ショートカット法などの攻撃に対して特別なリスクは見当たらない」ということであった。その後、更に情報公開が行われ、Skipjack アルゴリズムは 1998 年に公開された。[21]

(13) Khufu／Khafre

Khufu／Khafre は Xerox 社が開発した暗号で、マイクロプロセッサ上で非常に高速に処理が行えることが特長である。

内部のデータ処理アルゴリズムは同一であるが、256 個の S-box が Khafre では固定であり、Khufu は鍵から導出するという違いがある。Khufu／Khafre の *f* 関数は驚くほど単純であり、S-box による XOR 処理と８bit の左旋回シフト処理のみである。

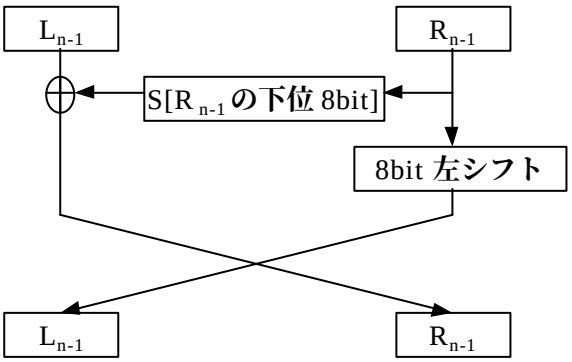


図 3.1-8　Khufu／Khafre の処理アルゴリズム

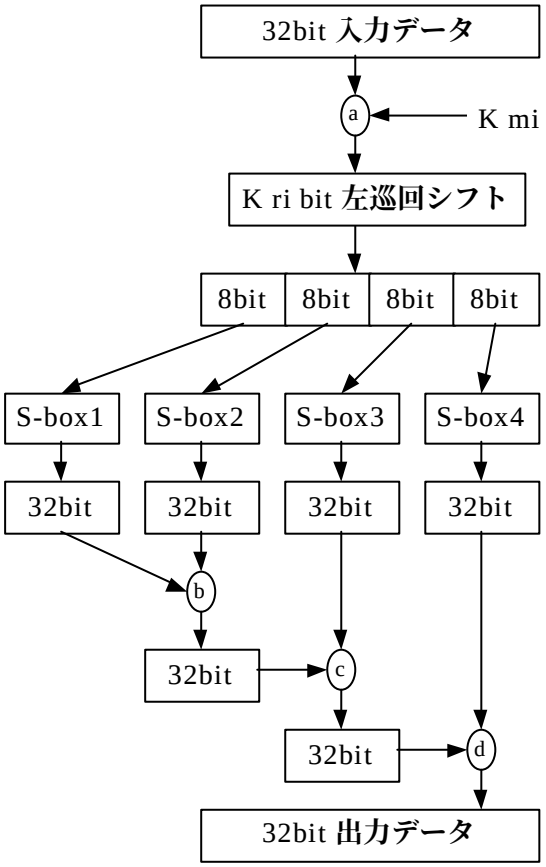
正式の仕様については未確認であるが、Khufu のソースコードといわれるものがイ

ンターネット上で入手可能である[22]。

(14) CAST

CAST は Entrust Technologies が開発した Feistel 型暗号で、各段の *f* 関数および鍵スケジュール処理を工夫することで DES よりも少ない段数で、より良い特性が得られるよう設計された。また、各パラメータは 32bit プロセッサでのソフトウェア処理に向くよう決定されている。

CAST の各段の処理は図 3.1-9 のようになっており、8 bit 入力－32bit 出力の S-box 変換と加算、減算、XOR のみからなっている。



CAST-128 の場合の各処理

TYPE	a	b	c	d
1	＋	XOR	－	＋
2	XOR	－	＋	XOR
3	－	＋	XOR	－

TYPE1 は 1,4,7,10,13,16 段目、
TYPE2 は 2,5,8,11,14 段目、
TYPE3 は 3,6,9,12,15 段目に使用する

図 3.1-9　CAST の基本処理

CAST は元々設計手法を指す語として使われ、ブロック長、鍵長、段数などが可変であるが、一般的に使用されている CAST 暗号は、64bit ブロック長、128bit 鍵長、16 段処理の CAST-128 と呼ばれるものである。この場合、サブ鍵として各段で 32bit の Km と 5 bit の Kr を用いる。CAST に関する資料は、インターネットの[23]、[24]より入手可能である。

CAST-128 をベースに、ブロック長 128bit、鍵長 128bit，192bit，256bit に対応した CAST-256 が AES に応募された[48]が、処理速度が遅く最終候補には残らなかった。

(15) GOST 28147-89

GOST 28147-89 は旧ソビエト連邦（現ロシア共和国）の政府標準暗号として、Zabotin，Glazkov，Isaeva らによって開発された暗号で、1989 年に制定された。

かつては、詳細なアルゴリズムなどまったく知られていなかったが、ソビエト連邦崩壊後は一転して商用利用に向けて積極的な PR 活動が行われているようである。もともとはロシア語の仕様書しかなく正確な翻訳が無かったが、Sun Microsystems の Diffie により英訳された仕様書[25]が現在入手可能である。よく GOST とだけ呼ばれるが、GOST はソビエト連邦における政府標準規格（GOsudorstvennyi STandard）のことであり、JIS や ANSI などと同じように多くの規格が存在する。この中の標準暗号が GOST 28147-89 である。

GOST 28147-89 の特長は、次のとおり。

- 32 段と段数が多い
- *f* 関数では mod32 の加算、11bit 左巡回シフト、および８つの４bit 入出力 S-box 変換のみからなる
- 鍵として 256bit のプライマリ鍵と 512bit のセカンダリ鍵が使用できる
- プライマリ鍵は８個の 32bit サブ鍵に分割され、昇順に３回、降順に１回使用される
- セカンダリ鍵は標準仕様には含まれないが順列テーブルとして８つの S-box を構成する
- ECB モードのほかに CFB，OFB モードが定義されている

なお、これまでに GOST 28147-89 の欠点として変異テーブルのセカンダリ鍵の定義が十分でない点、サブ鍵を合わせるとマスター鍵が求められるため鍵差分攻撃に弱い点が指摘されている。また、プライマリ鍵とセカンダリ鍵を合計すると 768bit となるが、B.Schneier の分析[26]によると、実効鍵長としては 610bit 相当である。

(16) CIPHERUNICORN-E

CIPHERUNICORN-E は NEC の角尾ら[27]によって 1998 年に発表されたブロック暗号で、鍵長 128bit、データ・ブロック長 64bit のブロック暗号である。CIPHERUNICORN-E は統計的な安全性が、暗号強度評価支援システムによって確認されていること[28]が特長である。

各段のデータ攪拌部は F 関数と呼ばれる 32bit 入出力の小さな処理を 16 段繰り返すインボリューション構造である。F 関数は４種類の 8bit 入出力 S-box から成る。S-box は、候補の中から線形／差分特性確率が理想値であり、２個または４個を組み合わせた場合に関して、8bit マスク以下で線形／差分特性確率が最小になるものを選択したとしている。初期/終了処理及び２段ごとに L 関数(64bit 入出力)を実行する。データ攪拌部の各段には、鍵スケジュール部が作成したサブ鍵を２個ずつ投入する。鍵スケジュール部は、入力データに依存して中間鍵を縮退させたサブ鍵を生成する。中間鍵を縮退させデータを攪拌するため、一対の既知平文から鍵を解読することが不可能となる。

NEC の暗号強度評価支援システムは、計算機実験によって暗号の優劣判定を行うデ

ータを採取し、３次元表示することにより評価結果の視認性を高めている。評価尺度は入出力の統計的な変化のみに着目するため、異なる暗号を共通の尺度で比較できる、としている。

(17) M8

M8 暗号[29]はブロック長 64bit、鍵長 64bit のブロック暗号で段数は可変である。M8 暗号では暗号化鍵のほかに、システム鍵と総称されるアルゴリズム決定鍵およびアルゴリズム拡張鍵によってアルゴリズムの細部を決定するという特徴がある。アルゴリズム決定鍵はデータ攪拌部および鍵スケジュール部の各段における基本処理での巡回シフト量などを規定する。アルゴリズム拡張鍵は 32bit 加算あるいは XOR される定数を与える。システム鍵の値によっては暗号強度が大きく変わることが示されている[30][31][32]。

以下では、ＡＥＳ第２ラウンド候補に残った５つのアルゴリズムについて説明する。

(18) MARS

MARS は、米国の IBM 社によって AES 候補として提案されたアルゴリズム[33]であり、ブロック長 128bit、鍵長は 32bit 単位で可変(128～1,248bit)である。32bit 汎用 CPU のソフトウェア実装での高速処理が可能になるように設計されている。

128bit ブロックを 32bit×４個のサブブロックに分割して変換する。データランダム化部は Type-3 Feistel 構造と呼ばれる構造をしており、４種類のラウンド関数を各 8 段、合計 32 段で構成される。

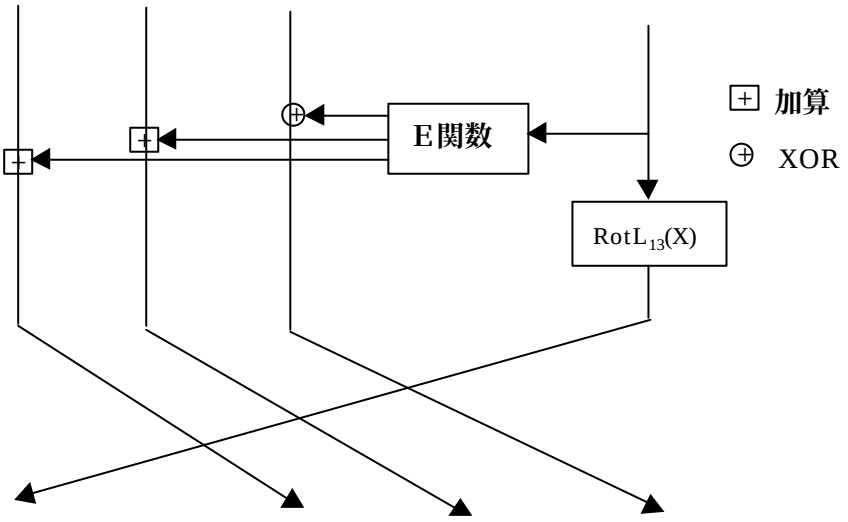


図 3.1-10　MARS(Cryptographic Core 部の type-3 Feistel 構造

最初の８段(forward mixing)と最後の８段(backwards mixing)は S-box(入力 9bit,出力 32bit)とデータ非依存型 bit シフトを中心に構成され、Cryptographic Core と呼ばれる中間の 16 段は S-box(入力 9bit,出力 32bit)やデータ依存型 bit シフト変換を含む E

関数によって構成されている。S-box は SHA-1 のアルゴリズムを基に生成し、差分特性や線形特性を考慮して設計されている。

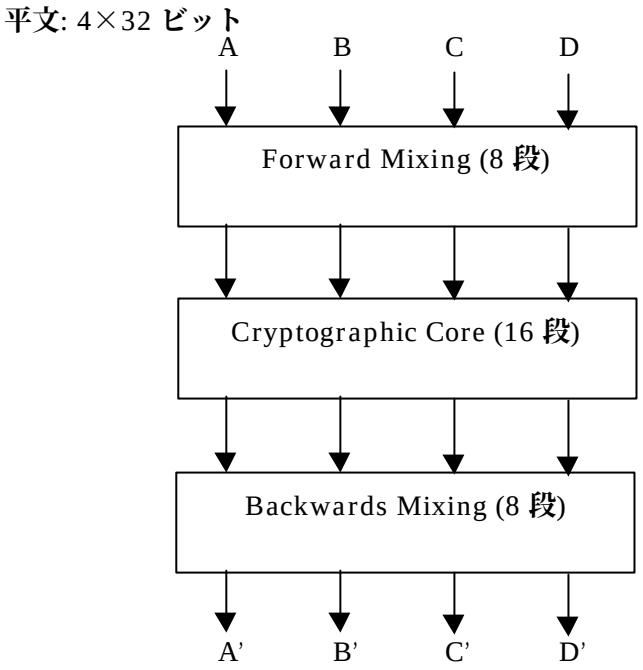


図 3.1-11 MARS のラウンド構成

32bit の乗算および加減算を多用しているため、32bit 乗算命令を持たないハードウェアで実装すると処理時間は長くなると言われている。

鍵スケジュール部はマスター鍵から 40 個の 32bit サブ鍵へ拡張し、サブ鍵の４～35 番までが Cryptographic Core の各段で２個ずつ利用される。マスター鍵を順次拡張してサブ鍵を生成する一般的な鍵拡張の後 Feistel により鍵の線形性を消す部分と、各段の乗算にそなえて必要に応じ内容を変更するための演算から構成される。

提案者による安全性分析結果[45]によると、16 段の差分特性確率は概ね2^{-240}、4 段の線形特性確率は概ね2^{-69}である。差分解読のためには2^{280}個以上の選択平文が必要であり、線形解読のためには2^{128}個以上の既知平文が必要である。また、弱鍵も見つかっていない。

(19) RC6

RC6 は MIT の Rivest 教授と RSA Laboratories が RC5 を AES の要求仕様を満たすように改良したアルゴリズムであり、ブロック長、鍵長(上限 2,040bit)、段数のいずれも可変である。MARS と同様に 32bit 乗算命令を備えた CPU でのソフトウェア実装が高速になるように設計されている。また、アルゴリズムの構造がシンプルであることも特徴である。

データランダム化部は RC5 と同様に Feistel 型で、データブロックを４つの等しい長さのサブブロックに分割した後、ラウンド関数に入力する。ラウンド関数はデータ

依存巡回シフト操作のほかに、サブブロック長を法とする加減乗算といった算術演算から構成されている。このため処理速度は実装プラットフォームに大きく依存し、ブロック長 128bit の場合、32bit の乗算命令やローテーション命令をサポートするプラットフォーム(たとえば Pentium II/Pro)では高速に実装できるが、そうでないプラットフォームでは著しく処理速度が低下する。スマートカードの８bitCPU への実装では Rijndael の 10 倍以上の処理時間を要するとの報告[35]がある。段数は可変であるが、ブロック長 128bit の場合には 20 段が推奨されている。

鍵スケジュール部には RC5 と同じアルゴリズムが使われている。すなわち、ユーザーが与える b bit の鍵をもとに、自然対数や黄金分割比より導かれたマジックナンバーを組み合わせ、各種演算をほどこして各段のサブ鍵が計算される。鍵スケジューリング部では、ラウンド関数の段数を r とすると、ブロック長と同じ長さの拡大鍵が(2r+4)個生成される[34]。

提案者による安全性分析結果[45]によると、18 段の最大差分特性確率は概ね2^{-264}であり、差分解読法に対して安全であるほか、線形解読法についても、少なくとも2^{182}個の既知平文が必要となるため安全であると述べている。

1999 年 Knudsen ら[36]は RC6 に対する新しい攻撃を提案し、彼らの Web ページで公開すると共に AES 第 2 ラウンドの一般コメントへ投稿した。これはc^2(カイ二乗)検定から得られる相関関係を利用した攻撃である。入力平文の一部を固定し、ラウンド関数におけるデータ依存巡回シフトの影響で、出力暗号文の一部に偏りが生じることを利用して入力側の拡大鍵の推定を行うというものである。15 段以下の RC6 の出力は乱数と区別できることが示され、ある種の弱鍵が存在する条件で 17 段までの非乱数性も示された。さらに 15 段までの RC6 に対して、全数探索よりも効率的な鍵推定アルゴリズムが示された。

(20) Rijndael

RIJNDAEL は、ベルギーの Daemen と Rijmen によってＡＥＳ候補に提案されたアルゴリズムであり、ブロック長・鍵長ともに 128，192，256 bit が利用可能であり、段数はブロック長および鍵長に依存して 10 段、12 段、14 段のいずれかとなる。例えばブロック長 128bit、鍵長 128bit の場合、10 段となる[37]。

Rijndael のラウンド関数は基本的に 8bit 単位で行われる。ラウンド関数は SPN 構造(Substitution-Permutation-Network)と呼ばれる構造をしている。線形変換層(bit シフト等)、非線形変換層(S-box による換字)、拡大鍵変換層(拡大鍵との排他的論理和)の３種類から構成されており、暗号化時は非線形変換層、線形変換層、拡大鍵変換層の順で変換が行われる。鍵スケジュール部にも、データランダム化部の bit シフトと換字変換が利用される。拡大鍵の長さはブロック長に等しく、(r+1)個生成される(r は段数)。

ラウンド関数の代表的な演算は、S-box における GF(2^8)の乗法逆元+アフィン変換、および GF(2^8)上の乗法を多用した線形変換であり、8bit CPU による簡単なブール演算の組合せで実現できる。このため、スマートカードでの実装において AES 候補中で最も処理時間が短いことが報告されている[35]、[38]。

提案者による安全性分析結果[45]によると、8段の最大差分特性確率は2^{-350}であり、差分解読法は適用不可能との評価。8段の最大線形特性確率は2^{-300}であり、線形解読法は適用不可能との評価。

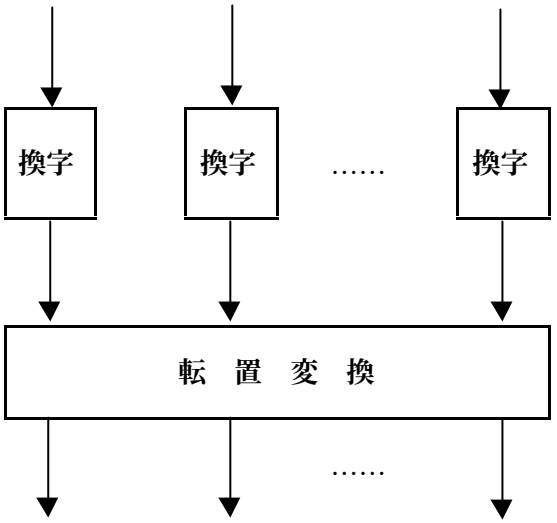


図3.1-12　SPN 構造

(21) Serpent

SERPENT は、イギリス Cambridge 大学の Anderson, イスラエル Technion の Biham , ノルウェーBergen 大学の Knudsen が AES 候補へ提案したアルゴリズム[39]であり、ブロック長が 128 bit 、鍵長が可変（上限 256 bit ）である。

データ攪拌部は、SPN(Substitution-Permutation Network)構造の 32 段ラウンド関数によって構成されている。鍵スケジュール部で 33 個の 128bit サブ鍵を生成して、128bit ブロックを暗号化する。32 ラウンドの前後に DES と同様の転置変換が行われ(Initial and Final Permutation）、各段では、入力ブロックをサブ鍵と XOR(排他的論理和)した後、S-box での置換と非線形変換（最終ラウンドを除く）が施される。

(Substitution)

S-box は 8 つあり、32 段の各々で 4 回づつ(1 段に 1 個の S-box）くり返し利用される。S-box は 4bit の入力を出力 4bit に非線形変換する。各段では一つの S-box が 32 個複製され、各々が入力 Text の 4bit づつを受け持つ。S-box は DES の S-box をもとに、一定の性質を持つまで S-box 内の数字同士をくり返し置換し生成される。DES の S-box を初期値に使うのは、S-box に（当初 DES で疑われたような）秘密の落とし戸が仕込まれていないことを明らかにするためという。

(Permutation)

最終ラウンドを除き、各ラウンドで S-box より出力された中間 Text は線形変換される。線形変換はデータ非依存型ローテーションとシフトと XOR の組み合わせから構成されている。S-box と線形変換の組み合わせの SPN により雪崩(avalanche)効果を最大化するよう設計されているという。

最終ラウンドでは、線形変換の代りにラウンド Key と XOT され S-box 変換されたものがさらに Sub Key (Key32）と XOR される。

Serpent の鍵スケジュール部では、入力されたマスター鍵をパディングして 256bit

に拡張し、それを word に分割して複数個 XOR して 132 個の Prekey を生成し、その後 S-box 変換して出力された 132 個の 32bit word を４つづつ用いてサブ鍵とする。

Serpent は、bit スライス(bitslice)手法による高速実装が可能となるように設計されている。bit スライス手法とは、Eli Biham[40]が FSE4(1997)で発表したブロック暗号の高速実装手法である。ブロック単位で暗号化を行う暗号を、１bit 単位の論理変換を使って実装し、複数ブロックを並列的に処理する高速化手法である。たとえば、32bit CPU を利用する場合、bit スライスでは 32 個のデータブロックを並列的に処理することが可能となる。Serpent は bit スライス実装のために 32bit 入出力の S-box を４種類用意している。

提案者による安全性分析結果[45]によると、24 段の場合、最大差分特性確率が2^{-232}以下、最大線形特性確率が2^{-109}以下になることから、差分解読法・線形解読法に対して安全である。また、５段の出力データのブール多項式次数が 243 程度になることから、高階差分攻撃に対しても安全である、という。

(22) Twofish

Twofish は米国・Counterpane Systems 社の Schneier，Kelsey，Hall，Ferguson，Hi/fn 社の Whiting ，UCB の Wagner によって AES 候補へ提案されたアルゴリズムであり、ブロック長は 128 bit 、鍵長は 128，192，256 bit が利用可能である

16 段の変形 Feistel 構造。各段ごとに入力テキストの左半分(64bit)が二つの 32bit 列に分けられて各々が f 関数に入力される。 f 関数の中で 32bit は更に四つの 8bit 列に分けられ、それぞれ別の鍵依存 S-box(8bit 入出力)で換字変換された後、四つあわせて GF(2^8)上の長さ４のベクトルとして GF(2^8)上の４×４行列（MDS：Maximum Distance Separable）との乗算(線形変換)を行う。

MDS から出力された２つの 32bit ベクトルは相互に加算され（擬似アダマール変換 PHT:Pseudo-Hadamard Transform： $a' = a + b \bmod 2^{32}$， $b' = a + 2b \bmod 2^{32}$）、さらにそれぞれがラウンドキーと加算された後、ラウンド入力 Text の右半分と XOR され、次ラウンドの入力（左半分）となる。その際 XOR の前後にそれぞれ１bit ローテーションが施される。この１bit ローテーションを除けば Feistel 型そのものである。

Twofish の鍵スケジュール部では、40 個の 32bit(1word)鍵と、f 関数の中で使われる４つの S-box が生成される。[41]

提案者による安全性分析結果[45]によると、差分解読法を適用するには７段の Twofish に対して少なくとも2^{131}個の選択平文が必要であることから、適用困難。12 段の Twofish に対して線形解読法を適用するには少なくとも2^{121}個の既知平文が必要であることから、適用困難。したがって 16 段の Twofish は差分解読法・線形解読法に対して十分な安全性を有しているという。Twofish の S-box は高次の非線形性を有していること等から、高階差分攻撃、補間攻撃、関連鍵攻撃に対しても高い安全性を有している、と分析している。

3.1.4 今後の展望 ～AES とその方向性～

これまでの多くのブロック暗号が採用している 64bit の鍵長／ブロック長は、32bit 汎用プロセッサ上での処理効率が非常に高くなることを利用している。これは、多くのアルゴリズムにおいて用いられているインボリューション（3.1.1 参照）では、入力を２（もしくは 2∩）個の bit 列に分割し処理を行っているためで、分割後のデータが 32bit や 16bit となり CPU や OS のワード長と一致するため、処理が容易になり速度が向上することによる。今後、CPU，OS ともに 64bit 化が予定されており、その時には 128bit の鍵長／ブロック長の暗号も速度向上が見込まれる。

一方、標準としての地位を築き上げた DES も、開発から 20 年以上経過し、仕様が最新のコンピュータにマッチしない上、鍵長／ブロック長が短く、コンピュータのコスト・パフォーマンス向上に伴い全数探索法が現実的な脅威となっている。近年登場したブロック暗号アルゴリズムの中には、より長い鍵長や、可変の鍵長に対応し、特に米国の輸出規制を容易にクリアできるようなものも登場し始めた。DES 自体も、３つの鍵を用いる Triple-DES 方式で、さまざまな分野への適用が図られている。

(1) Triple-DES

後述の AES が標準化され広く一般に普及するまでの間、DES の後継として最も有力視されているのが、DES を３回繰返す Triple-DES である[42]。ANSI X9.52[43]及び NIST による FIPS PUB46-3[4]として DES に代わる標準暗号として認定された。Triple-DES による暗号化は、異なる２つまたは３つの鍵で、DES の暗号化 - 復号 - 暗号化の順に３回繰返す方式であり、鍵長を 112bit または 168bit に伸ばすことにより、全数探索法に対する安全性を向上させている。また、DES と同じブロック長(64bit)であり、３回とも同じ鍵を用いる場合には、従来の DES と同じ結果が得られることから、DES を利用したシステムからの移行が比較的容易である。一方、DES の処理を３回繰り返すことから、暗号化および復号に要する時間はそれだけ長くなる。

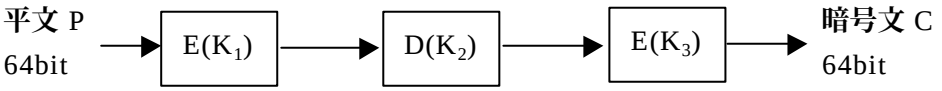


図 3.1-13 Triple-DES による暗号化(TECB モード)

図 3.1-13 で、$E(K_i)$は DES による暗号化を $D(K_i)$は DES による復号処理を表わす。

3.1.5 節で述べる利用モードで、暗号化に $C = E_{K3}(D_{K2}(E_{K1}(P)))$ を、

復号に $C = D_{K1}(E_{K2}(D_{K3}(P)))$ を用いることにより、従来の DES 利用モードを、そのまま Triple-DES にも適用することができる。K1,K2,K3 の３つの鍵がすべて異なる場合を 3-key Triple-DES、K1=K3 かつ K1 K2 の場合を 2-key Triple-DES と呼ぶ。３つの鍵をすべて同じ(K1=K2=K3)にすれば、従来の DES と同じ処理結果が得られるため、DES を前提とした既存システムで Triple-DES へ移行することは比較的容易である。

2-key Triple-DES は鍵長が 112bit のため、全数探索法では 2^{112} の計算量が必要である。3-key Triple-DES は鍵長が 168bit だから、全数探索法では 2^{168} の計算量が必要である。しかし、中間一致による選択平文攻撃(Merkle and Hellman)によると、2-key

Triple-DES では2^{57}、3-key Triple-DES では2^{112}と、全数探索法に対して大幅に計算量を削減可能であるとしている[42]。その他、Oorschot と Wiener による既知平文攻撃も提案されているが、これらは膨大な外部記憶媒体容量が必要であること、膨大な情報量を収集しなければならないこと、などから現実の脅威となる可能性は低いと考えられている[42]。

Triple-DES はブロック長が DES と同じ 64bit であるため、暗号文一致攻撃(Matching Ciphertext Attack)や辞書攻撃(Dictionary Attack)に対する安全性は DES と変わらない。このため、インナーCBC モードや CBCM モード等の利用モードが提案されたが、これらの利用モードに対しても攻撃法が提案され、根本的な対策にはブロック長を長くすることが求められる。

このような状況下で、1997 年1月、NIST（米国商務省 標準化局）は、DES に替わる新たな暗号の標準を模索し、AES（Advanced Encryption Standard）として、一般の意見を取り入れながら公募プログラムを開始、1997 年9月 12 日に正式の応募要綱を発表した[44]。

(2) AES の概要

- DES の後継となるブロック暗号方式の確立
- 一般からの意見を参考に、公募形式で進める
- 2000 年をめどに標準化完了し 2020 年から 30 年程度の利用を前提とする

(3) AES 候補アルゴリズムの必要条件

- 共通鍵暗号であること
- ブロック暗号であること
- 少なくとも鍵長－ブロック長が以下の組み合わせが可能なこと
- 128-128bit、192-128bit、256-128bit

(4) アルゴリズム評価基準

① 安全性

i. 同一鍵長・ブロック長での他のアルゴリズムとの比較

ii. 平文のランダム化したものと暗号文の類似性比較

iii. 数学的安全性の根拠

iv. 評価プロセス中に指摘される安全性に関する問題

② コスト

i. ライセンス要件

ii. 計算効率性

iii. 実装時必要なメモリ量（ハードウェアはゲート数）

③ その他アルゴリズムの特長

i. 各種アプリケーション対応の柔軟性

ii. ハードウェア、ソフトウェアとの親和性

iii. シンプルな構造

(5) AES 標準化スケジュール

① 1997 年１月 2 日上記要件の草案と意見募集

② 1997 年４月 15 日概要説明と収集意見に対するディスカッションのためのワークショップ開催

③ 1997 年９月 12 日募集要項発表

④ 1998 年６月 15 日応募締切
（1998 年４月 15 日までの応募に対しては５月 15 日までに必要な修正内容が提示される）

⑤ レビューの上、全ての応募内容(15 件)を公開し意見収集

⑥ 1998 年８月 20 日～22 日 第１回 AES カンファレンス開催(Crypt'98 併催)
応募者による説明と質疑応答、意見収集

⑦ 技術評価第１ラウンド
カンファレンスの結果レビュー、候補の絞り込み

⑧ 1999 年３月 22,23 日、第２回 AES カンファレンス開催(FSE 併催)
NIST による技術評価内容の議論、より一層の絞り込み

⑨ 1999 年８月９日 技術評価第２ラウンド候補の発表
安全性、計算効率や知的所有権などを考慮し最大５つまで候補の絞り込み
MARS, RC6, Rijndael, Serpent, Twofish の５件に絞られた。

⑩ 2000 年４月 13,14 日 第３回 AES 候補カンファレンス開催(ニューヨーク,FSE 併催)

⑪ 2000 年５月 15 日 技術評価第２ラウンド終了

⑫ 2001 年夏 NIST による最終選定

DES の FIPS 見直しの行われた 1999 年までに AES の結論は出ないため、FRB（米国連邦準備基金）は、AES 案制定までの間 Triple-DES を使用できるよう求めている。その結果、Triples-DES は FIPS46-3 として米国標準に制定された。一方、DES から AES へのスムーズな移行を求める声も多く、今後暗号応用システムを開発する立場からすると、両方に対応が容易なシステム設計が必要となると思われる。但しブロック長が DES, Triple-DES は 64bit であるのに対して、AES は 128bit であることから、既存システムからのスムーズな移行を疑問視する意見もある。なお、現時点では強力な暗号アルゴリズムを組み込んだ製品については、米国政府の輸出規制の対象となっている(7.4.1 参照)。

もともと DES に代わるということは、「機密扱い（軍用など）でない重要な情報の暗号化」に用いるべきものであるため、公開できる程度の安全性である可能性も否定はできない。ただし、商用化の上では標準化されることによる爆発的な普及が DES のように起きるかどうかも疑問視されている。これは、DES 開発当時に比べ様々な暗号方式の選択肢が非常に広くなってきたことも関係する。今後は、コストと安全性を十分検討した上でいろいろな選択肢の中から暗号アルゴリズムを選ぶということが一般化するかもしれない。

3.1.5 ブロック暗号方式の利用モード

ブロック暗号方式は、同一の鍵と平文の組み合わせでは常に同じ暗号文が得られる。したがって頻度の高いコードや金額など、推定されやすいデータを繰返し含む平文の暗号化には向かない。

ISO および NIST では DES の利用モードとして（図 3.1-14）、これらの欠点を補う利用モードを標準化している[6][46]。

① ECB モード（Electronic Code Book）

オリジナルの DES 利用モード。

② CBC モード（Cipher Block Chaining）

平文の暗号化前に、前のブロックの暗号文を XOR してから暗号化を行う。

同一の平文であっても、ランダムな暗号文を XOR するため異なる暗号結果が得られる。

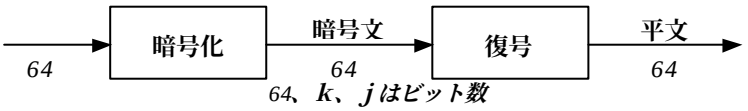
なお、最初のブロックの暗号化には初期値（または初期化ベクタ）（IV）と呼ばれる値を XOR する。復号には暗号化鍵と IV が必要である。

③ CFB モード（Cipher Feedback Mode）

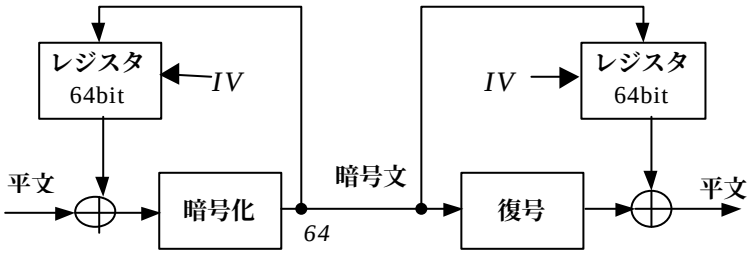
④ OFB モード（Output Feedback Mode）

上記③、④の２方式は、DES 暗号化を平文に対してではなく、レジスタの内容に対して行う。平文の暗号化は、得られた DES 暗号結果のランダムな出力 bit ストリームと XOR することによりなされるため、一種のストリーム暗号方式である。すなわち、DES の bit 撹乱・ランダム化による非線形変換を、乱数発生器として使用するものである。レジスタには、CBC モードと同様の初期値（IV）をセットする。復号には暗号化鍵と IV が必要である。これら２方式は、誤り訂正効果がある、暗号化・復号いずれも DES の暗号化ロジックですむという長所を持つものの、処理速度が極端に低下するという欠点もあわせもつことが特徴である。したがって、比較的低速な回線の暗号化や、専用チップを用いる暗号化に向いている。

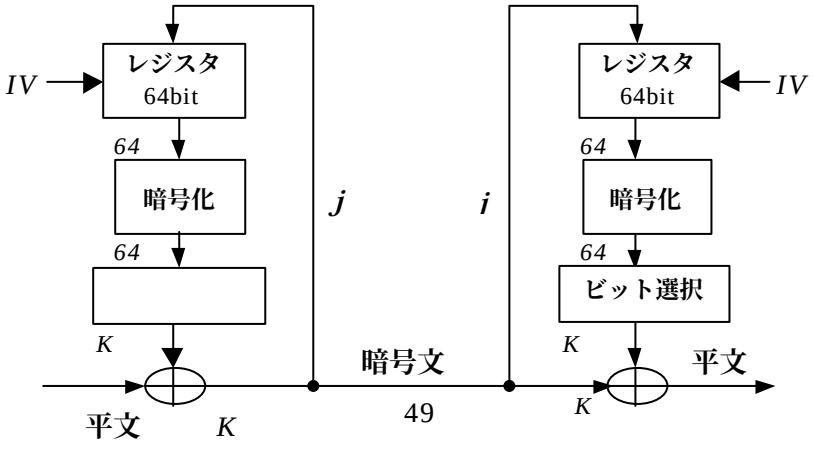
(a) ECB モード



(b) CBC モード



(c) CFB モード



(d)　OFB コード

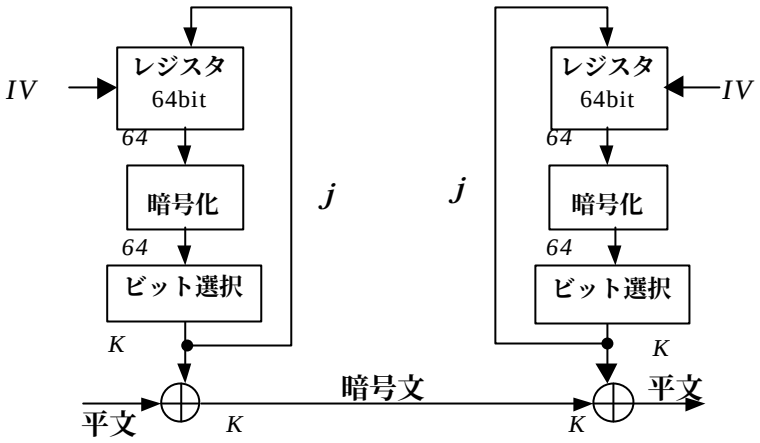
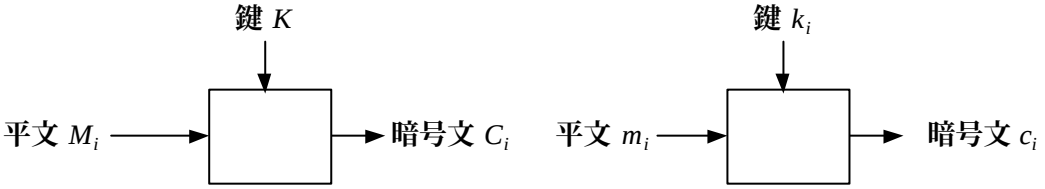


図 3.1-14　利用モード

3.1.6　共通鍵ストリーム暗号の定義と原理

ストリーム暗号とは、平文を 1 bit ないし 1 Byte 単位に、ある特定の鍵系列により逐次的に暗号化する手法であり、逐次暗号とも呼ばれる。



ブロック暗号
鍵 K は**固定**
$M = M_1\ M_2\ M_3\ \cdots$
M_i は数十 bit

ストリーム暗号
鍵 k_i は**逐次変化**
$M = m_1\ m_2\ m_3\ \cdots$
m_i は 1 ～数 bit

図 3.1-15　ブロック暗号とストリーム暗号

最も単純なストリーム暗号は、十分に長い乱数系列を用意し、暗号化と復号に際して平文の n Bit 目と乱数列の n Bit 目を XOR するもので、一般に**バーナム暗号**と呼ばれている。現在のところ、暗号化と復号には同一の鍵系列を用いるものしかなく、あえて共通鍵をつけて呼ばれることはないので、ここでは以下ストリーム暗号と呼ぶ。

ストリーム暗号では、平文と暗号文の対応が分かれば、用いている鍵系列が直ちに求まる。そのため、安全性は主としてこの鍵系列として用いる乱数の性質に依存する。一方で、乱数列の生成は一般にブロック暗号の攪拌処理に比べて高速であり、ソフトウェアのみ用いた場合でも高速な通信経路に追随する処理速度が得られる。完全に使い捨ての乱数列を用いたバーナム暗号は、鍵系列となる乱数列が秘匿される限り、理論的に解読不可能であるが、鍵系列の長さが平文と同じ物になるためどのようにこれを生成、配送、保管するかが問題となる。

一般的に使用されるストリーム暗号は、特定の長さの鍵を入力とする乱数発生器の出力を鍵系列として用いる。乱数発生の種々の方式については 3.4 で詳細に述べるので、ここで

は乱数の望ましい性質として以下の点が重要であることを述べるにとどめる。

① ０、１のなど頻度性（０と１の出現確率がそれぞれ 0.5 に近いこと）
② 長周期性（繰り返しの発生周期が非常に長いこと）
③ 非線形性（乱数発生に非線形操作を含んでいること）
④ 無相関性（乱数発生器の入出力間および出力間に相関が無いこと）
⑤ 線形複雑度が大（線形フィードバックレジスタ表現の場合の bit 数が長いこと）

また、歴史的にも性能的にもストリーム暗号の適用分野は音声通話やデータ通信時の暗号方式として用いられることが多く、そのため音声欠落やデータの再送を極力押さえるため、自己同期方式を用いることがある。3.1.5 で述べた CFB モードや OFB モードも自己同期ストリーム暗号方式である。

3.1.7 各種ストリーム暗号の比較

ストリーム暗号には、ブロック暗号方式の DES のように普及した方式が無く、一般的に知られている種類も少ない。ここでは比較的良く使われている方式について若干の説明を行う。

(1) RC4

RC4 は他の RC シリーズと同様に、Rivest により 1987 年に開発されたストリーム暗号方式である。鍵長は 2,048bit までの可変長である。

RC4 のアルゴリズムは秘密とされてきたが、現在では同等のソースコードの入手が可能である[47]。ただし、北米においては RC4 の商標権を RSA 社が保持しているため、インターネット上などでは ARCFOUR(A(not)-RC4）という名称で用いられることもある。RC2 と同様に、可変長の鍵を用いることが出来るため、米国の輸出規制を逃れることができ、Lotus Notes など多くの製品にインプリメントされている。RC4 では標準で８×８の８bit S-box を乱数発生器として用い、OFB モードで動作する[26]。

乱数発生ロジックは以下のように非常に簡単な物となっている。

$i = (i + 1) \bmod 256$
$j = (j + S_i) \bmod 256$
swap S_i S_j
$t = (S_i + S_j) \bmod 256$
乱数 $K = S_t$

この K を平文（または暗号文）と XOR することで暗号化（復号）を行う。S-box の初期化は、鍵を用いて以下のロジックを実行することで行われる。鍵はあらかじめ $K_0, K_1 \cdots K_{255}$ の配列に必要回数繰り返し格納される。

for $i = 0$ to 255
$j = (j + S_i + K_i) \bmod 256$
swap S_i S_j

上記の処理は８bit の RC4 のものであるが、16×16 の 16bit S-box を用いるものも定義が可能である。この場合、65,536 個の S-box の初期化のために $K_0, \cdots K_{65535}$ の鍵を用いることが可能で、鍵長は最大１Mbit にも及ぶが、初期化処理に必要な時間は増大する。一方、処理の単位が２Byte 単位となるため処理速度は高速化できると考えられる。

(2) SEAL

SEAL は IBM の Rogaway と（DES の開発者の一人）Coppersmith により開発された方式で、米国で特許が成立している。

SEAL は 32bit プロセッサに最適化したアルゴリズムを持ち、きわめて高速に動作する（486-50MHz でおよそ 58Mbps）[26]。SEAL の特長は、SHA アルゴリズムを用いて、160bit の鍵 k と 32bit の n を用いて 64kByte 以下の擬似乱数列を生成することである。そのために、SEAL では 9×32 個の 64bit S-box を用いるが、そのためのセットアップ時間が 200 個の SHA ハッシュ値の計算量に匹敵するため、少量のデータの暗号化には向かないといわれる。

(3) カオス暗号

カオス（Chaos）とは本来「混沌」を意味するギリシャ語であるが、常微分方程式や差分方程式で記述される物理的な統計確率現象の、離散時間における実数値解の不規則な振る舞いを表わすのものとして用いられる。

アナログカオスには、特定の条件下で初期条件が異なっていても、十分時間が経過した後の定常状態ではほぼ同一の振る舞いをするという同期現象がある。これを利用したのが一般的な「カオス暗号」であるが、カオスのパラメータそのものが構造的安定性を有しているため推定されやすいという欠点がある。また、アナログ値を近似して用いるため微少な誤差による同期外れなども懸念される。

デジタルカオスは、アナログ回路の常微分方程式や差分方程式を、デジタル計算で解く手法を利用するため、アナログカオスに比べて再現性が高い。近年、デジタルカオスを用いたと称する暗号がいくつか提唱されているが、パラメータの取り方や安全性解析など、明らかにされていないケースが多い。

現時点では、このような暗号では様々な攻撃に対しての安全性が証明されておらず、提唱者、実装者による仕様の公開と、安全性に対する評価が必要であろう。カオス自体は、エルゴード性（任意の初期値から出発した軌道が任意の点の近傍を何度も通過する）を持つため、軌道上の特定の点から初期値を求めることは極めて困難であり、初期値を鍵とする暗号では各種解読法に対する強さが期待できる。

今後、カオスの持つ様々な性質がより深く解析され、実証的にも原理的にも安全な暗号方式が提唱されることを期待したい。

3.2 公開鍵暗号

公開鍵暗号は鍵生成・公開鍵処理・秘密鍵処理のアルゴリズムの組によって構成される。鍵生成は公開鍵と呼ばれる公開関数を規定するパラメータと秘密鍵と呼ばれる秘密関数を規定するパラメータの対を生成する。公開鍵は第三者に公開しても構わないが、秘密鍵は対応する公開鍵がバインドされたエンティティのみが秘密裡に保持する。また、公開鍵とエンティティのバインドが保証されなければならない。通常、証明書を用いることによってこの問題を解決しており、鍵生成に引き続いて鍵の正当性を保証する証明書の発行という形で鍵登録が行われる。

公開鍵処理は主に公開鍵をパラメータとする公開関数の計算アルゴリズムで構成される。

（関数対の満たす性質によっては乱数生成や補助的な一方向性関数の計算アルゴリズムも用いられる。）秘密鍵処理は主に秘密鍵をパラメータとする秘密関数の計算アルゴリズムで構成される（関数対の満たす性質によっては乱数生成や補助的な一方向性関数の計算アルゴリズムも用いられる。）。

公開鍵暗号の興味深い点は公開関数と秘密関数との関係が既知であるにも関わらず、秘密鍵を知らない限り秘密関数の計算を行うことが計算量的に難しいことであり、これを実現するトリックは数学的な問題によっている。ただし、秘密関数の計算を行うことが計算量的に困難なことが証明されているわけではなく、ある意味で公開鍵暗号という技術は計算量についての我々の無知に基づいたものと言える。

3.2.1 実現機能による分類

公開鍵暗号は公開関数と秘密関数の対がどのような関係を満たすかによって実現できる機能が分類できる。関数対の満たす関係として基本的なのは以下の８タイプである（表 3.2-1 参照）。ここでπは公開関数をσは秘密関数をφは補助的な一方向性関数を表している。また、c は秘密関数への入力(チャレンジ)をr は秘密関数の出力(レスポンス)をk は使い捨てられるランダムな鍵を表している。

表3.2-1 公開鍵暗号の実現できる機能による分類

	機能	関数対のみたす関係式	方式
(1)	暗号(メッセージ守秘)	$\sigma(\pi(r)) = r$	RSA
(2)		$\sigma(\pi(k,r)) = r$	OU, ElGamal 暗号, OAEP, EPOC-1, PSEC-1, Cramer-Shoup
(3)	鍵共有	$\pi(k) = \sigma(\varphi(k))$	DH
(4)	メッセージ復元署名	$\pi(\sigma(c)) = c$	RSA
(5)		$\pi(\sigma(k,c)) = c$	ESIGN, NR
(6)	署名	$\pi(c,\sigma(c)) = 1$	RSA 署名
(7)		$\pi(c,\sigma(k,c)) = 1$	ESIGN, Schnorr 署名, GQ 署名, ElGamal 署名, DSA 署名, ESIGN 署名, NR 署名
(8)	２交信相手認証	上記の全て	*認証
(9)	３交信相手認証	$\pi(\varphi(k),c,\sigma(k,c)) = 1$	Schnorr 認証, GQ 認証（ElGamal 認証, DSA 認証）

関数対のみたす性質を簡略に図示すると以下のようになる（図 3.2-1 参照）。ここでボックスが関数であり、矢印はデータの入出力をb は検証結果である真偽値を表している。

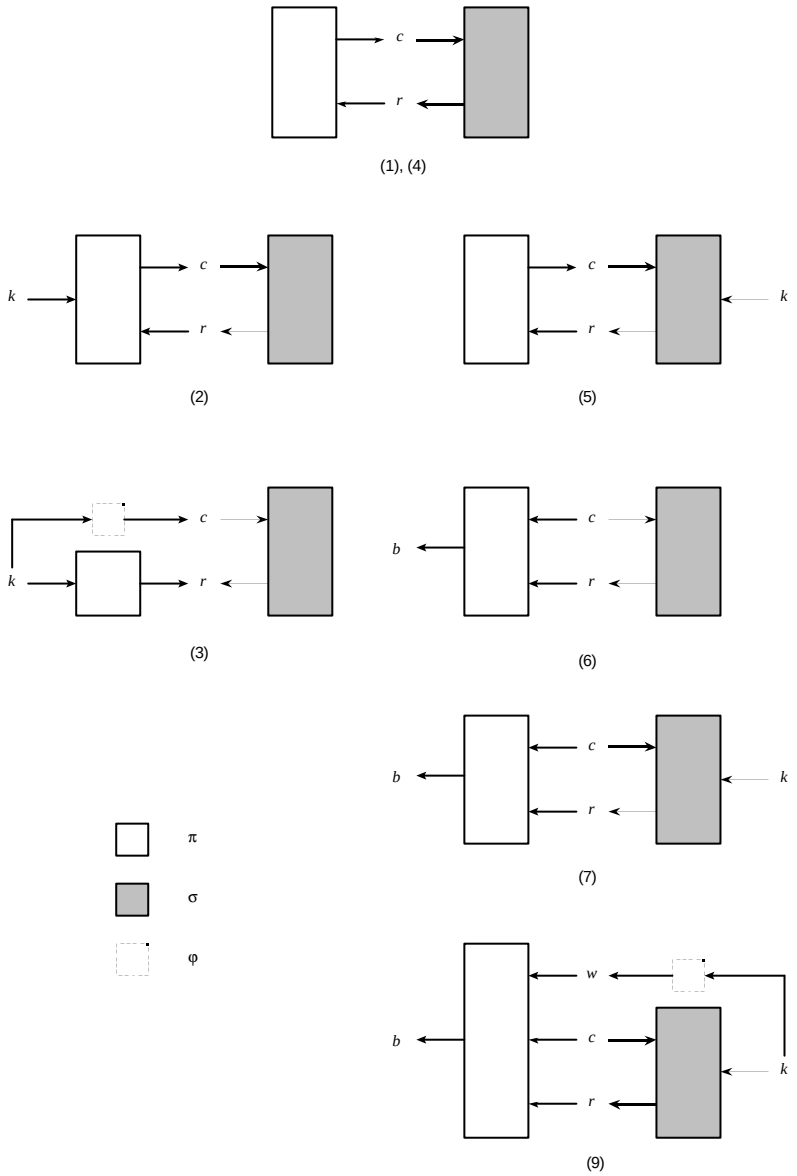


図 3.2-1　関数対の関係

以下、各機能毎に公開鍵処理と秘密鍵処理がどのように構成されるかを説明する。

(1)　暗号方式

関数対[5]

$\pi : R \to C$

$\sigma : C \to R$

が $\sigma(\pi(r)) = r$ **を満たし、また、与えられた** $c \in \mathrm{Im}\,\pi$ [6] **より** $c = \pi(r)$ **を満たす** $r \in R$ **を見つけ**

---

[5] σ は必ずしもいたるところで定義されていなくても良い。σ がいたるところで定義されていると能動的攻撃に対して脆弱な場合がある。

[6] 写像 $f : X \to Y$ に対して $\mathrm{Im}\, f \coloneqq \{f(x); x \in X\}$ を表す。

ることが難しい[7]場合、以下のような暗号(メッセージ守秘)方式が構成できる。ここで r はメッセージ、c は暗号文である。

公開鍵処理（暗号化）	秘密鍵処理（復号）
1. メッセージ r を入力する 2. 暗号文 $c \leftarrow \pi(r)$ を計算する 3. 暗号文 c を出力する	1. 暗号文 c を入力する 2. メッセージ $r \leftarrow \sigma(c)$ を計算する 3. メッセージ r を出力する

暗号化は公開情報のみから行えるのでメッセージ r があらかじめ小さな部分集合 $R' \subset R$ に属していることが分かれば[8]、R' に属するメッセージを順次暗号化し、解読対象の暗号文に一致するメッセージを探索する一種の選択平文攻撃を行えるので注意しなければならない。

(2) **確率的暗号方式**

関数対[5]

$$\pi : K \times R \to C$$
$$\sigma : C \to R$$

が[9] $\sigma(\pi(k,r)) = r$ を満たし、また、与えられた $c \in \mathrm{Im}\,\pi$ よりある $k \in K$ に対して $c = \pi(k,r)$ を満たす $r \in R$ を見つけることが難しい[7]場合、以下のような確率的暗号(メッセージ守秘)方式が構成できる。ここで r はメッセージ、c は暗号文である。

公開鍵処理（暗号化）	秘密鍵処理（復号）
1. メッセージ r を入力する 2. 乱数 k を生成する 3. 暗号文 $c \leftarrow \pi(k,r)$ を計算する 4. 暗号文 c を出力する	1. 暗号文 c を入力する 2. メッセージ $r \leftarrow \sigma(c)$ を計算する 3. メッセージ r を出力する

公開鍵処理で乱数生成が行われており、同一のメッセージ r に対しても異なる暗号文 c が生成されるので、乱数空間 K が十分大きければ、(1)で述べた選択平文攻撃に対して安全性が高い。

---

[7] 攻撃（暗号解読）者の立場で完全な r を見つけることを**完全解読**と言い、r のなんらかの部分情報を見つけることを**部分解読**と言う。任意の c に対して完全解読が困難ならば関数対は**一方向性**を持つと言い、任意の c に対していかなる部分解読も困難ならば関数対は**強秘匿性**を持つと言う。また c の解読にあたって、秘密関数 σ を実行するオラクルを $c' \neq c$ に利用できる場合を**適応的選択暗号文攻撃**と言う。**オラクル**とは内部動作を抽象した純粋なブラックボックスのことだと考えれば良い。

[8] 例えば、メッセージとして共通鍵暗号の鍵（通常公開鍵暗号のメッセージ空間よりもずっと小さい）を定型のフォーマットでパディングして暗号化するような場合。

[9] 集合 X, Y に対して $X \times Y := \{(x, y);\ x \in X,\ y \in Y\}$ を表す。

(3) **鍵共有方式**

関数対

$\pi: K \to R$

$\sigma: C \to R$

$\varphi: K \to C$

が $\delta(k) = \sigma(\varphi(k))$ **を満たし、また、与えられた** $c \in \mathrm{Im}\varphi$ **よりある** $k \in K$ **に対して** $c = \varphi(k)$ **かつ** $r = \pi(k)$ **を満たす** $r \in R$ **を見つけることが難しい場合、以下のような鍵共有方式が構成できる。ここで** c **は中間鍵、** r **は共有鍵である。公開鍵保持者は中間鍵を秘密鍵保持者に送り、秘密鍵保持者は中間鍵より共有鍵を生成する。**

公開鍵処理（共有鍵生成）	秘密鍵処理（共有鍵生成）
1. 乱数 k を生成する 2. 中間鍵 $c \leftarrow \varphi(k)$ を計算する 3. 中間鍵 c を出力する 4. 共有鍵 $r \leftarrow \pi(k)$ を計算する 5. 共有鍵 r を出力する	1. 中間鍵 c を入力する 2. 共有鍵 $r \leftarrow \sigma(c)$ を計算する 3. 共有鍵 r を出力する

(4) **メッセージ復元署名方式**

関数対[10]

$\pi: R \to C$

$\sigma: C \to R$

が $\pi(\sigma(c)) = c$ **を満たし、また、与えられた** $c \in C$ **より** $c = \pi(r)$ **を満たす** $r \in R$ **を見つけることが難しい**[11]**場合、以下のようなメッセージ復元署名方式が構成できる。ここで** c **はメッセージ、** r **は署名値である。**

公開鍵処理（メッセージ復元）	秘密鍵処理（署名生成）
1. 署名値 r を入力する 2. メッセージ $c \leftarrow \pi(r)$ を計算する 3. メッセージ c を出力する	1. メッセージ c を入力する 2. 署名値 $r \leftarrow \sigma(c)$ を計算する 3. 署名値 r を出力する

(5) **確率的メッセージ復元署名方式**

関数対[10]

$\pi: R \to C$

$\sigma: K \times C \to R$

が $\pi(\sigma(k, c)) = c$ **を満たし、また、与えられた** $c \in C$ **より** $c = \pi(r)$ **を満たす** $r \in R$ **を見つけ**

---

[10] π は必ずしもいたるところで定義されていなくても良い。π がいたるところで定義されていれば存在的偽造が可能なことに注意しよう。

[11] 攻撃（署名偽造）者の立場で任意の c に対して r を見つけることを**一般的偽造**と言い、ある c に対して r を見つけることを**存在的偽造**と言う。また c の偽造にあたって、$c \notin C'$ に対応する $(r_{c'})_{c' \in C'}$（ここで $r_{c'}$ は c' に対応した署名値であるとする）を利用できる場合を**既知文書攻撃**と言い、秘密関数 σ を実行するオラクルを $c' \neq c$ に利用できる場合を**適応的選択文書攻撃**と言う。

ることが難しい[11]場合、以下のようなメッセージ復元署名方式が構成できる。ここでcはメッセージ、rは署名値である。

公開鍵処理（メッセージ復元）	秘密鍵処理（署名生成）
1. 署名値rを入力する 2. メッセージ$c \leftarrow \pi(r)$を計算する 3. メッセージcを出力する	1. メッセージcを入力する 2. 乱数kを生成する 3. 署名値$r \leftarrow \sigma(k,c)$を計算する 4. 署名値rを出力する

公開鍵処理で乱数生成が行われており、同一のメッセージcに対しても異なる署名値rが生成される。

(6) 署名方式

関数対

$\pi : C \times R \rightarrow \{0,1\}$

$\sigma : C \rightarrow R$

が$\pi(c,\sigma(c)) = 1$を満たし、また、$\pi(c,r) = 1$を満たす$(c,r) \in C \times R$を見つけることが難しい[11]場合、以下のような署名方式が構成できる。ここでcはメッセージ、rは署名値である。

公開鍵処理（署名検証）	秘密鍵処理（署名生成）
1. メッセージcを入力する 2. 署名値rを入力する 3. 検証結果$b \leftarrow \pi(c,r)$を計算する 4. 検証結果bを出力する	1. メッセージcを入力する 2. 署名値$r \leftarrow \sigma(c)$として計算する 3. 署名値rを出力する

(7) 確率的署名方式

関数対

$\pi : C \times R \rightarrow \{0,1\}$

$\sigma : K \times C \rightarrow R$

が$\pi(c,\sigma(k,c)) = 1$を満たし、また、$\pi(c,r) = 1$を満たす$(c,r) \in C \times R$を見つけることが難しい[11]場合、以下のような確率的署名方式が構成できる。ここでcはメッセージ、rは署名値である。

公開鍵処理（署名検証）	秘密鍵処理（署名生成）
1. メッセージcを入力する 2. 署名値rを入力する 3. 検証結果$b \leftarrow \pi(c,r)$を計算する 4. 検証結果bを出力する	1. メッセージcを入力する 2. 乱数kを生成する 3. 署名値$r \leftarrow \sigma(k,c)$を計算する 4. 署名値rを出力する

公開鍵処理で乱数生成が行われており、同一のメッセージcに対しても異なる署名値rが生成される。

(8) **２交信認証方式**

今まで述べてきた(1)ないし(7)の性質を満たす関数対を用いて２交信認証方式を構成することができる。ただし、これらの構成が可能なのは秘密関数σを実行するオラクルによる攻撃に対して耐性を持つ場合に限られる。以下の例ではcはチャレンジ、rはレスポンスである。公開鍵保持者は秘密鍵保持者にランダムなチャレンジを送信し、受信したレスポンスが正しいことを確認することによって認証を行う。

	公開鍵処理（検証）	秘密鍵処理（証明）
(1)	1. 参照値r'をランダムに生成する 2. チャレンジ$c \leftarrow \pi(r')$を計算する 3. チャレンジcを出力する 4. レスポンスrを入力する 5. 検証結果$b \leftarrow (r = r')$を求める 6. 検証結果bを出力する	1. チャレンジcを入力する 2. レスポンス$r \leftarrow \sigma(c)$を計算する 3. レスポンスrを出力する
(2)	1. 参照値r'をランダムに生成する 2. 乱数kを生成する 3. チャレンジ$c \leftarrow \pi(k, r')$を計算する 4. チャレンジcを出力する 5. レスポンスrを入力する 6. 検証結果$b \leftarrow (r = r')$を求める 7. 検証結果bを出力する	
(3)	1. 乱数kを生成する 2. 参照値$r' \leftarrow \pi(k)$を計算する 3. チャレンジ$c \leftarrow \varphi(k)$を計算する 4. チャレンジcを出力する 5. レスポンスrを入力する 6. 検証結果$b \leftarrow (r = r')$を求める 7. 検証結果bを出力する	

	公開鍵処理（検証）	秘密鍵処理（証明）
(4)	1. チャレンジcをランダムに生成する 2. チャレンジcを出力する 3. レスポンスrを入力する 4. 検証結果$b \leftarrow (c = \pi(r))$を計算する 5. 検証結果$b$を出力する	1. チャレンジcを入力する 2. レスポンス$r \leftarrow \sigma(c)$を計算する 3. レスポンスrを出力する
(6)	1. チャレンジcをランダムに生成する 2. チャレンジcを出力する 3. レスポンスrを入力する 4. 検証結果$b \leftarrow \pi(c, r)$を計算する 5. 検証結果bを出力する	

	公開鍵処理（検証）	秘密鍵処理（証明）
(5)	1. チャレンジ c をランダムに生成する 2. チャレンジ c を出力する 3. レスポンス r を入力する 4. 検証結果 $b \leftarrow (c = \pi(r))$ を計算する 5. 検証結果 b を出力する	1. チャレンジ c を入力する 2. 乱数 k を生成する 3. レスポンス $r \leftarrow \sigma(k, c)$ を計算する 4. レスポンス r を出力する
(7)	1. チャレンジ c をランダムに生成する 2. チャレンジ c を出力する 3. レスポンス r を入力する 4. 検証結果 $b \leftarrow \pi(c, r)$ を計算する 5. 検証結果 b を出力する	

(9) **3交信認証方式**

関数対

$\pi : W \times C \times R \to \{0,1\}$

$\sigma : K \times C \to R$

$\varphi : K \to W$

が $\pi(\varphi(k), c, \sigma(k,c)) = 1$ を満たし、また、与えられた $c \in C$ より $\pi(w,c,r) = 1$ を満たす $r \in R$ を見つけることが易しいような $w \in W$ を見つけることが難しい場合、以下のような 3 交信認証方式が構成できる。ここで w はウィットネス[12]、c はチャレンジ、r はレスポンスである。公開鍵保持者は秘密鍵保持者からウィットネスを受信した後にチャレンジを送信し、受信したレスポンスが正しいことを確認することによって認証を行う。

公開鍵処理（検証）	秘密鍵処理（証明）
1. ウィットネス w を入力する 2. チャレンジ c をランダムに生成する 3. チャレンジ c を出力する 4. レスポンス r を入力する 5. 検証結果 b を $b \leftarrow \pi(w,c,r)$ として計算する 6. 検証結果 b を出力する	1. 乱数 k を生成する 2. ウィットネス w を $w \leftarrow \varphi(k)$ として計算する 3. ウィットネス w を出力する 4. チャレンジ c を入力する 5. レスポンス r を $r \leftarrow \sigma(k,c)$ として計算する

このような関数対はより基本的な関数対

$\pi' : C \times R \to W$

$\sigma : K \times C \to R$

$\varphi : K \to W$

[12] ウィットネスとは、秘密情報自体（この場合乱数 k）を開示せずに提示される秘密情報の証拠情報であり、ウィットネス提示後に秘密情報を変更することを不可能にするものである。

で $\pi'(c,\sigma(k,c))=\varphi(k)$ を満たし、また、与えられた $c\in C$ より $\pi'(c,r)=w$ を満たす $r\in R$ を見つけることが易しいような $w\in W$ を見つけることが難しいものから、関数 π を

$\delta(w,c,r)=b$
$w'\leftarrow\delta'(c,r)$ $b\leftarrow(w=w')$

と構成することによって得られる。

3.2.2 安全性の根拠による分類と鍵生成

公開鍵暗号は公開鍵から秘密鍵を導出することの難しさがどのような問題を根拠にしているかによって分類できる。現在、実用的な公開鍵暗号が安全性の根拠としている問題は以下の 2 タイプである（表 3.2-2 参照）。

問題	内容	方式
素因数分解問題	合成数 n からその素因数 p を求める	RSA, OU, ESIGN, GQ
離散対数問題	$y=g^x$ [$y=xg$]から指数[係数] x を求める[13] （ここで G は有限群[14]、 $g\in G$ ）	DH, Schnorr, ElGamal, DSA

表 3.2-2 安全性の根拠による分類

素因数分解が困難な合成数は、通常、大きな素数を生成してそれらの積を求めることによって作られる。また、現在、実用に供されている離散対数問題が困難な有限群としては、有限体[15]上の乗法群[16]および楕円曲線[17]であり、表 3.2-3 に現れるパラメータを生成するため

---

[13] 本稿では、一般に「 $A[A']...B[B']...$ 」は「 $A...B...$ 、または $A'...B'...$ 」を表すことにする。ここでは、普通、乗法群上では乗法的な記法が、楕円曲線上では加法的な記法が用いられるので両方を併記した。

[14] 群とは、結合法則を満たし中立元と対称元を持つ算法の定義された集合であり、有限群とは位数（元の個数）が有限な群のことである。群の算法は xy のように乗法的に書かれる場合と $x+y$ のように加法的に書かれる場合がある（加法的に書かれる場合は可換法則 $x+y=y+x$ が仮定され、可換群という）。乗法的[加法的]な場合、中立元は単位元[零]と呼ばれ $1[0]$ と書かれ、対称元は逆元[反元]と呼ばれ $x^{-1}[-x]$ のように書かれる。

[15] 体とは、可換群の構造を与える加法と、可換群の構造を与える乗法を同時に備えた集合であり、有限体とは位数が有限な体のことである。

[16] 体 K の乗法群とは、 0 を除いた集合 $K^\times:=K-\{0\}$ を乗法に関する群と見なしたものである。

[17] 体 K 上の楕円曲線 E とは、平面上の非特異 3 次曲線（標数が 2 や 3 でなければ $y^2=x^3+ax+b$ で右辺が重根を持たないものと考えれば良い）に特殊な点 0（平面上の点ではないので無限遠点と呼ばれる）を加えた集合に $x,y,z\in E$ が直線 L と楕円曲線 E との交点に一致する（つまり $E\cap L=\{x,y,z\}$ ）ときに $x+y+z=0$ となるように加法を定義して構成される群である（本当にこのようにして群の構造が定義される。興味があれば参考文献[6]を見よ。）

には、

- 素数生成
- 有限体生成
- 楕円曲線生成

が要求される。素数生成は、乱数生成に素数判定法（Fermat テスト・Solovay-Strassen テスト・Miller-Rabin テストといった確率的アルゴリズムと Jacobi 和テスト・Atkin テストといった確定的アルゴリズムがある。これらのアルゴリズムについては参考文献[5]を見よ）を組み合わせて実現される。有限体生成については、以前は乗法群しか用いられていなかったこともあって素体しか用いられなかった[18]のでほとんど素数生成に同義であったが、離散対数問題のベースとして楕円曲線が注目されてからは、標数２の体も多く扱われ、さらに最近では OEF(Optimal Extension Field)と呼ばれる CPU の自然な演算サイズの標数を持った拡大体が注目されている。非素体における体要素のデータ構造（つまり、基底の取り方）やそれと密接に関係する体算法処理の高速化については多くの研究がなされている。楕円曲線生成については、生成された楕円曲線の位数の計算が問題であり、当初は位数の計算が不要な超特異（トレースが 0）や anomalous（トレースが 1）な曲線が提案されてきたが、1993 年に Menezes, Okamoto, Vanstone による MOV 帰着（楕円曲線を定義体の小さな拡大体の乗法群に埋め込む）、1997 年に Semaev, Smart, Satoh, Araki による SSSA 攻撃（楕円曲線と拡大体の加法群との効率的な同型を与える）が発見されて暗号学的な価値が消滅した。一方楕円曲線の位数を数え上げる Schoof のアルゴリズムの改良（Atkin-Elkies アルゴリズム）と計算機の高速化は、ランダムに生成した楕円曲線を用いることを十分実用的にしている。

以下に基本的な関数対の計算式のパラメータ（つまり公開鍵と秘密鍵）を列挙する（表 3.2-3 参照）。ここで、$p,q,n,e,d \in \mathbf{Z}$, $\varepsilon,\gamma,u,v \in \mathbf{Z}/n\mathbf{Z}$, $g,y \in G$, $x,\delta \in \mathbf{F}_p$ であり、表中かっこでくくられたパラメータは、必ずしも各エンティティ固有でなくともよい複数のエンティティが共通に利用可能なパラメータである。

	π	σ	φ
RSA	$n\,(,e)$	n,d	
OU	n,ε,γ	p,δ	
ESIGN	$n\,(,e)$	$p,q,n\,(,e)$	
GQ	$(n,p,)\,v$	$(n,)\,u$	(n,p)
DH	$(G,)\,y$	$(G,)\,x$	(G,g)
Schnorr	$(G,g,)\,y$	$(G,p,)\,x$	(G,g)

[18] 有限な素体とはある素数 p を法とした剰余演算の定義された集合 $\mathbf{F}_p := \{0,1,\kappa\ ,p-1\}$ のことであり、任意の有限体 K は必ずある素数 p で決まる素体 $\mathbf{F}_p$ を部分体として含む（のでその文脈で拡大体と呼ばれる）。また、このとき p を K の標数という。拡大体の乗法群における離散対数問題を基礎体の乗法群の離散対数問題に能率的に帰着させる Coppersmith のアルゴリズムがある。

表3.2-3 基本方式の鍵パラメータ

(1) RSA

整数eを定めておく。公開鍵処理の高速化のためにはeの値は小さいほうが有利である一方で、$e=3$のように極端に小さな場合は攻撃法が知られており（例えば、参考文献[10]）、$e=2^{16}+1$が良く用いられる。

RSA 鍵生成
1. 素数p, qを$p-1[q-1]$がeと互いに素になるようにランダムに生成する
2. $n \leftarrow pq$を計算する
3. $\lambda(n) \leftarrow \mathrm{lcm}(p-1, q-1)$を計算する
4. $d \leftarrow e^{-1} \bmod \lambda(n)$を計算する
5. nを公開鍵として出力する
6. (n, d)を秘密鍵として出力する

(2) Okamoto-Uchiyama

鍵生成で用いられる関数

$$L_p(x) := \frac{x^{p-1} - 1 \bmod p^2}{p} \in \mathbf{F}_p$$

はいわゆる Fermat 商と呼ばれる関数であり、乗法群$\left\{x \in \mathbf{Z}/p^2\mathbf{Z};\ x \bmod p = 1\right\}$から加法群$\mathbf{F}_p$への同型[19]を与える。

Okamoto-Uchiyama 鍵生成
1. 素数p, qをランダムに生成する
2. $n \leftarrow p^2 q$を計算する
3. $u, \gamma \in \mathbf{Z}/n\mathbf{Z}$をランダムに生成する
4. $\delta \leftarrow L_p(\gamma)^{-1} \bmod p$を計算する
5. $\varepsilon \leftarrow u^n \bmod n$を計算する
6. (n, ε, γ)を公開鍵として出力する
7. (p, δ)を秘密鍵として出力する

(3) ESIGN

整数eを定めておく。RSA の鍵生成と同様に、(1)と同様にeが小さいほうが処理の

[19] 一般に同型とは構造を保存する全単射（１体１の対応）であり、ここでは、群構造を問題にしているので構造の保存とは$L(xy) = L(x) + L(y)$を満たすこと（乗法が加法に形を保ったまま移行している）。

高速化の点で有利となる。$e = 2^{10}$程度が推奨されている。

ESIGN 鍵生成
1. 素数 p, q をランダムに生成する
2. $n \leftarrow p^2 q$ を計算する
3. n を公開鍵として出力する
4. n, p, q を秘密鍵として出力する

(4) Guillou-Quisquater

素因数分解が困難な合成数nと素数pを定めておく[20]。(1)ないし(3)の場合とは異なり、複数のエンティティが共通の(n, p)を利用することができる。

Guillou-Quisquater 鍵生成
1. $u \in \mathbf{Z}/n\mathbf{Z}$ をランダムに生成する
2. $v \leftarrow u^p \bmod n$ を計算する
3. v を公開鍵として出力する
4. u を秘密鍵として出力する

(5) Diffie-Hellman, Schnorr

離散対数問題が困難な群Gと位数がpの元$g \in G$を定めておく。(4)と同様に複数のエンティティが(G, g, p)を利用することができる。

Diffie-Hellman, Schnorr 鍵生成
1. $x \in \mathbf{F}_p$ をランダムに生成する
2. $y \leftarrow g^x \ [xg]$ を計算する
3. y を公開鍵として出力する
4. x を秘密鍵として出力する

3.2.3 基本的な関数対

ここでは最も基本的な公開鍵暗号の関数対を計算するアルゴリズムを列挙して説明する。以下に基本的な関数対の計算式を列挙する（表 3.2-4 参照）。$c \in C, r \in R, k \in K, w \in W$であり、他のパラメータは3.2.2の表 3.2-3 で挙げられたものである。

	π	σ	φ	C	R	K	W
RSA	r^e	c^d		$\mathbf{Z}/n\mathbf{Z}$	$\mathbf{Z}/n\mathbf{Z}$		
OU	$\varepsilon^k \gamma^r$	$L_p(c)\delta$		$\mathbf{Z}/n\mathbf{Z}$	$\mathbf{F}_p$	I_n	
ESIGN	r^e	$S_{p,q,n,e}(k,c)$		$\mathbf{Z}/n\mathbf{Z}$	$\mathbf{Z}/n\mathbf{Z}$	I_{pq}	
GQ	$v^c r^p$	ku^{-c}	k^p		$\mathbf{Z}/n\mathbf{Z}$	$\mathbf{Z}/n\mathbf{Z}$	$\mathbf{Z}/n\mathbf{Z}$
DH	$y^k \ [ky]$	$c^x \ [xc]$	$g^k \ [kg]$	G	G	$\mathbf{F}_p$	
Schnorr	$y^c g^r \ [cy + rg]$	$k - cx$	$g^k \ [kg]$	$\mathbf{F}_p$	$\mathbf{F}_p$	$\mathbf{F}_p$	G

表 3.2-4　関数対の計算式

$$S_{p,q,n,e}(k,c) \coloneqq k + \left(\left\lfloor \frac{c - k^e \bmod n}{pq} \right\rfloor \left(ek^{e-1} \right)^{-1} \bmod p \right) pq$$

(1) RSA

RSA は 1978 年に Rivest,Shamir と Adleman によって発見され、暗号方式とメッセージ復元署名方式を同時に与える関数対[21]

$\pi : R \to C$

$\sigma : C \to R$

を持つ。

(2) Okamoto-Uchiyama

Okamoto-Uchiyama は 1998 年に岡本龍明と内山成憲によって発見され、確率的暗号方式を与える関数対

$\pi : K \times R \to C$

$\sigma : C \to R$

を持つ。素因数分解問題の困難さを仮定すると受動的攻撃に対して一方向性を持つことが証明されており、大きな特徴になっている[22]。

(3) Diffie-Hellman

Diffie-Hellman は 1976 年に Diffie と Hellman によって発見され、鍵共有方式を与える関数対

$\pi : K \to R$

$\sigma : C \to R$

$\varphi : K \to C$

を持つ。

(4) ESIGN

ESIGN は 1990 年に岡本龍明によって発見された関数対

$\pi : R \to C$

$\sigma : K \times C \to R$

であり、

$\sigma'(k, c') = r$	$\pi'(r) = c'$
$c \leftarrow c' \cdot 2^{2\lfloor n/3 \rfloor}$ $r \leftarrow \sigma(k, c)$	$c \leftarrow \pi(r)$ $c' \leftarrow \left\lfloor c / 2^{2\lfloor n/3 \rfloor} \right\rfloor$

---

[20] ここで p は n の因数と言うわけではない。

[21] このような関数対の公開関数を落し戸付き**一方向性置換**と言う。

[22] さらに p 部分群問題を仮定すると受動的攻撃に対して強秘匿であることが証明されている。

とすることによって確率的メッセージ復元署名方式を与える関数対

$$\sigma' : K \times C' \to R$$
$$\pi' : R \to C'$$
$$\left(C' := I_{2^{|n|/3}}\right)$$

に変換することができる。ここで、簡単のためにnの bit 長を3の倍数とした。

(5) **Schnorr**

1991 年に Schnorr によって発見され、3 交信認証方式を与える関数対

$$\pi : C \times R \to W$$
$$\sigma : K \times C \to R$$
$$\varphi : K \to W$$

を持つ。

(6) **ElGamal**

1985 年に ElGamal によって発見され、ElGamal 署名方式の原型の 3 交信認証方式を与える関数対

$$\pi : W \times C \times R \to W$$
$$\sigma : K \times W \times K \to R$$
$$\varphi : K \to W$$

を持つ。Schnorr 関数対

$$\pi : C \times R \to W$$
$$\sigma : K \times C \to R$$
$$\varphi : K \to W$$

から

$\sigma'(k', w', k) = r'$	$\pi'(w', k, r') = w$
$c \leftarrow -\rho(w')$	$c \leftarrow \rho(w')/r'$
$r \leftarrow \sigma(k, c)$	$r \leftarrow k/r'$
$r' \leftarrow r/k'$	$w \leftarrow \pi(c, r)$

と変換することによって得られる。ここで$\rho : G \to \mathbf{F}_p$ は写像である。

(7) **DSA**

FIPS-186 で標準化されている DSA 署名方式の原型であり、ElGamal を若干変形して得られる 3 交信認証方式を与える関数対

$$\pi : W \times C \times R \to W$$
$$\sigma : K \times W \times K \to R$$
$$\varphi : K \to W$$

を持つ。Schnorr 関数対

$\pi : C \times R \to W$

$\sigma : K \times C \to R$

$\varphi : K \to W$

から

$\sigma'(k', w', k) = r'$	$\pi'(w', k, r') = w''$	$\varphi'(k) = w''$
$c \leftarrow -w'$ $r \leftarrow \sigma(k, c)$ $r' \leftarrow r / k'$	$c \leftarrow w' / r'$ $r \leftarrow k / r'$ $w \leftarrow \pi(c, r)$ $w'' \leftarrow \rho(w)$	$w \leftarrow \varphi(k)$ $w'' \leftarrow \rho(w)$

と変換することによって得られる。ここで $\rho : G \to \mathbf{F}_p$ は写像である。

(8) Guillou-Quisquater

Guillou-Quisquater は 1988 年に Guillou と Quisquater によって発見され、3 交信認証方式を与える関数対

$\pi : C \times R \to W$

$\sigma : K \times C \to R$

$\varphi : K \to W$

を持つ。

3.2.4 初等的な変換

ここでは関数対の型を変換する初等的な方法について具体例を交えて解説する。

(1) ElGamal 暗号

一般に鍵共有方式を与える関数対

$\pi : K \to R$

$\sigma : C \to R$

$\varphi : K \to C$

は、任意の共通鍵暗号

$E : R \times R' \to R'$

（ここで R は鍵空間、R' はメッセージ空間）を用いて、

$\pi'(k, r') = c'$	$\sigma'(c') = r'$
$c \leftarrow \varphi(k)$ $r \leftarrow \pi(k)$ $C \leftarrow E_r(r')$ $c' \leftarrow (c, C)$	$(c, C) \leftarrow c'$ $r \leftarrow \sigma(c)$ $r' \leftarrow E_r^{-1}(C)$

とすることによって、確率的暗号方式を与える関数対

$\pi' : R \times R' \to C'$

$\sigma' : C' \to R'$

$(C' := C \times R')$

に変換できる。Diffie-Hellman にこの変換を施して得られるのが ElGamal 暗号である。

(2) Nyberg-Rueppel

一般に 3 交信認証方式の原型を与える関数対

$\pi : C \times R \to W$

$\sigma : K \times C \to R$

$\varphi : K \to W$

は、任意の共通鍵暗号

$E : W \times C \to C$

（ここで W は鍵空間、C はメッセージ空間）を用いて、

$\sigma'(k, c') = r'$	$\pi'(r') = c'$
$w \leftarrow \varphi(k)$	$(c, r) \leftarrow r'$
$c \leftarrow E_w(c')$	$w \leftarrow \pi(c, r)$
$r \leftarrow \sigma(k, c)$	$c' \leftarrow E_w^{-1}(c)$
$r' \leftarrow (c, r)$	

とすることによって、確率的メッセージ復元署名方式を与える関数対

$\sigma' : K \times C \to R'$

$\pi' : R' \to C$

$(R' := C \times R)$

に変換できる。Schnorr にこの変換を施して得られるのが Nyberg-Rueppel である。

(3) RSA 署名

一般にメッセージ復元署名方式を与える関数対

$\pi : R \to C$

$\sigma : C \to R$

は、任意のハッシュ関数 $H : C' \to C$ を用いて、

$\sigma'(c') = r$	$\pi'(c', r) = b$
$c \leftarrow H(c')$	$c \leftarrow H(c')$
$r \leftarrow \sigma(c)$	$b \leftarrow \left(c = \pi(r)\right)$

とすることによって、署名方式を与える関数対

$\sigma' : C' \to R$

$\pi' : C' \times R \to \{0,1\}$

に変換できる。RSA にこの変換を施して得られるのが RSA 署名である。

(4) ESIGN 署名, Nyberg-Rueppel 署名

一般に確率的メッセージ復元署名方式を与える関数対

$\pi : R \to C$

$\sigma : C \to R$

は、任意のハッシュ関数 $H : C' \to C$ を用いて、

$\sigma'(k,c')=r$	$\pi'(c',r)=b$
$c \leftarrow H(c')$ $r \leftarrow \sigma(k,c)$	$c \leftarrow H(c')$ $b \leftarrow (c=\pi(r))$

とすることによって、確率的署名方式を与える関数対

$\sigma' : C' \to R$

$\pi' : C' \times R \to \{0,1\}$

に変換できる。ESIGN にこの変換を施して得られるのが ESIGN 署名であり、Nyberg-Rueppel にこの変換を施して得られるのが Nyberg-Rueppel 署名である。

(5) Schnorr 署名, Guillou-Quisquater 署名

一般に 3 交信認証方式を与える関数対

$\pi : C \times R \to W$

$\sigma : K \times C \to R$

$\varphi : K \to W$

は、任意のハッシュ関数 $H : W \times C' \to C$ を用いて、

$\sigma'(k,c')=r'$	$\pi'(c',r')=b$
$w \leftarrow \varphi(k)$ $c \leftarrow H(w,c')$ $r \leftarrow \sigma(k,c)$ $r' \leftarrow (w,r)$	$(w,r) \leftarrow r'$ $c \leftarrow H(w,c')$ $b \leftarrow (w=\pi(c,r))$

とすることによって、確率的署名方式を与える関数対

$\sigma' : K \times C' \to R$

$\pi' : C' \times R \to \{0,1\}$

$(R' := W \times R)$

に変換できる。Schnorr にこの変換を施して得られるのが Schnorr 署名であり、Guillou-Quisquater にこの変換を施して得られるのが Guillou-Quisquater 署名である。

(6) ElGamal 署名, DSA 署名

一般に 3 交信認証方式の原型を与える関数対

$\pi : W \times C \times R \to W$

$\sigma : K \times C \to R$

$\varphi : K \to W$

は、任意のハッシュ関数 $H : C' \to C$ を用いて、

$\sigma'(k,c')=r'$	$\pi'(c',r')=b$
$w \leftarrow \varphi(k)$ $c \leftarrow H(w,c')$ $r \leftarrow \sigma(k,c)$ $r' \leftarrow (w,r)$	$(w,r) \leftarrow r'$ $c \leftarrow H(w,c')$ $b \leftarrow (w=\pi(c,r))$

とすることによって、確率的署名方式を与える関数対

$\sigma' : K \times C' \to R$
$\pi' : C' \times R \to \{0,1\}$
$(R' := W \times R)$

に変換できる。ElGamal にこの変換を施して得られるのが ElGamal 署名であり、DSA にこの変換を施して得られるのが DSA 署名である。

3.2.5 証明可能安全性を付与する変換

ここではランダムオラクルモデルや汎用一方向性関数の存在の基に、関数対の型を能動的攻撃に対して証明可能安全性を付与するように変換する方法について具体例を交えて解説する。

ランダムオラクルモデルとは、暗号を構成する際には理想的なランダム関数[23]を用いて安全性を証明し、実現する際に理想的なランダム関数を実用的なハッシュ関数で置き換える方法論である。

(1) OAEP

1994 年に Bellare と Rogaway によってランダムオラクルモデルの基に、落し戸付き一方向性置換を与える関数対を、適応的選択暗号文攻撃に対して強秘匿性を持った確率的暗号方式を与える関数対に変換する手法が発見された。

関数対

$$p : R \to C$$
$$s : C \to R$$

はランダム関数

$H : K \to R' \times T$
$G : R' \times T \to K'$
$(K \times R' \times T := R)$

を用いて、

[23] 理想的なランダム関数とは、理想的な乱数表で定義されるような関数だと考えれば良い。理想的な乱数表は全くランダムなはずなのでそのような関数を実現するためには可能な全ての入力に対する巨大な表を用意するしかなく全く現実的ではない理想的な仮定である。これに対して、実用的なハッシュ関数は全ての入力に対する表とは比較にならないほど小さなアルゴリズム記述によって実現されている。

$\pi'(k,r')=c$	$\sigma'(c)=r'$
$r''\leftarrow(r',t_0)\oplus H(k)$ $k'\leftarrow k\oplus G(r'')$ $r\leftarrow(k',r'')$ $c\leftarrow\pi(r)$	$r\leftarrow\sigma(c)$ $(k',r'')\leftarrow r$ $k\leftarrow k'\oplus G(r'')$ $(r',t)\leftarrow r''\oplus H(k)$ if $t\neq t_0$ then reject

とすることによって、確率的暗号方式を与える関数対

$\pi':K\times R'\to C$

$\sigma':C\to R'$

に変換できる。元になった関数対が落し戸付き一方向性置換ならば、変換して得られる関数対は適応的選択暗号文攻撃に対して強秘匿性を持った確率的暗号方式を与えることが、H,G がランダム関数だという仮定の基に証明できる。

RSA にこの変換を施して得られるのが OAEP である。したがって、OAEP は RSA 問題が困難[24]だとすれば、ランダムオラクルモデルの基で非常に強い安全性を持った暗号だということができる。

(2) EPOC-1，PSEC-1

OAEP では変換の対象となる関数対は一方向性置換しか許していない。一方、楕円曲線暗号などで用いられる離散対数系の関数対には一方向性置換はないので、OAEP の手法は適用できない。

1998 年に藤崎英一郎と岡本龍明によって、ランダムオラクルモデルの基に、一方向性を持つ確率的暗号方式を与える関数対

$\pi:K\times R\to C$

$\sigma:C\to R$

をランダム関数 $H:R\to K$ を用いて、

$\pi'(k',r')=c$	$\sigma'(c)=r'$
$r\leftarrow(k',r')$ $k\leftarrow H(r)$ $c\leftarrow\pi(k,r)$	$r\leftarrow\sigma(c)$ $k\leftarrow H(r)$ $(k',r')\leftarrow r$ if $c\neq\pi(k,r)$ then reject

とすることによって、適応的選択暗号文攻撃に対して強秘匿性を持つ確率的暗号方式を与える関数対

$\pi:K'\times R'\to C$

$\sigma:C\to R'$

$(K'\times R'\coloneqq R)$

---

[24] ほとんど同語反復だが、RSA 問題が困難というのは RSA 暗号が一方向性を持つということである。

に変換する手法が発見された。

Okamoto-Uchiyama にこの変換を施して得られるのが EPOC-1 であり、ElGamal 暗号にこの変換を施して得られるのが PSEC-1 である。

(3) EPOC-2，PSEC-2

1999 年に藤崎英一郎と岡本龍明によってランダムオラクルモデルの基に、一方向性を持つ確率的暗号方式を与える関数対

$\pi : K \times R \to C$

$\sigma : C \to R$

をランダム関数

$H : R \times R' \to K$

$G : R \to SK$

と共通鍵暗号 $E : SK \times R' \to R'$ を用いて、

$\pi'(r, r') = c'$	$\sigma'(c') = r'$
$k \leftarrow H(r, r')$	$(c, C) \leftarrow c'$
$c \leftarrow \pi(k, r)$	$r \leftarrow \sigma(c)$
$K \leftarrow G(r)$	$K \leftarrow G(r)$
$C \leftarrow E_K(r')$	$r' \leftarrow E_K^{-1}(C)$
$c' \leftarrow (c, C)$	$k \leftarrow H(r, r')$
	if $c \neq \pi(k, r)$
	then reject

とすることによって、適応的選択暗号文攻撃に対して強秘匿性を持つ確率的暗号方式を与える関数対

$\pi' : R \times R' \to C'$

$\sigma' : C' \to R'$

$(C' := C \times R')$

に変換する手法が発見された。この変換で興味深い点は、共通鍵暗号 E との組み合わせも込みにして証明可能安全性が得られることである。実際の公開鍵暗号による暗号方式は、もっぱら共通鍵暗号の鍵配送のためにのみ使われ、実際のデータは配送される共通鍵で暗号化されることが普通であり、そのような現実の運用の安全性を保証できるのは好ましいといえるだろう。

Okamoto-Uchiyama にこの変換を施して得られるのが EPOC-2 であり、ElGamal 暗号にこの変換を施して得られるのが PSEC-2 である。

(4) Cramer-Shoup

1998 年に Cramer と Shoup によって発見された手法は、(1)ないし(3)で述べた手法と比較してランダムオラクルモデルを仮定しない点が際立っている。実際、彼等の手法では汎用一方向性関数の存在のみを仮定している。一方、これまでの方式が関数対の汎用的な変換手法だったのに対して、Cramer-Shoup は ElGamal 暗号にしか適用できない。Cramer-Shoup の安全性証明の方法論では、Diffie-Hellman 問題の困難さ（つまり、ElGamal 暗号の一方向性）よりさらに強い Diffie-Hellman 決定問題が困難であ

るという仮定に基づいて安全性の証明を与えている。

ElGamal 暗号の鍵生成において、位数が p の元 $h \in G$ を追加して定めておき、鍵生成に以下の処理を追加する。

Cramer-Shoup 鍵生成
5. $x_0, x_1 \in \mathbf{F}_p$ をランダムに生成する
6. $y_0 \leftarrow g^{x_0} h^{x_1}$ $[x_0 g + x_1 h]$を計算する
7. $y_1 \leftarrow g^{x_2} h^{x_3}$ $[x_2 g + x_3 h]$を計算する
8. y_0, y_1 を公開鍵として出力する
9. x_0, x_1 を秘密鍵として出力する

また、一方向性関数 $H : C \times G \to \mathbf{F}_p$ を用いて

$\pi'(k, r) = c'$	$\sigma'(c') = r$
$c \leftarrow \pi(k, r)$	$(c, d, \alpha, \beta) \leftarrow c'$
$d \leftarrow h^k$	$r \leftarrow \sigma(c)$
$\alpha \leftarrow H(c, d)$	$(\chi, C) \leftarrow c$
$\beta \leftarrow (y_0 y_1^{\alpha})^k$	$\beta' \leftarrow \chi^{x_0 + \alpha x_1} d^{x_2 + \alpha x_3}$
$c' \leftarrow (c, d, \alpha, \beta)$	if $\beta \neq \beta'$ then reject

とすることによって、Diffie-Hellman 決定問題[25]が困難であるという仮定の基で、適応的選択暗号文攻撃に対して強秘匿な確率的暗号方式

$\pi' : K \times R \to C'$

$\sigma' : C' \to R$

$\left(C' := C \times G \times \mathbf{F}_p \times G\right)$

に変換できる。

3.2.6 今後の展望

楕円曲線を用いた公開鍵暗号では、鍵サイズを理想的に小さくできることからパフォーマンスの向上等様々な利点があり実システムでもより多く用いられることになる可能性があり、標準化活動も活発に行われている（[11][12][13]）。楕円曲線上の公開鍵暗号は、従来の RSA や Diffie-Hellman, DSA 等に比較して鍵生成のバラエティが多様であり、多くの研究も進展している。安全性の観点からこれらの研究の動向を良く把握する必要がある。

近年、PKCS#1 version 1.5 に対する Bleichenbacher 攻撃や ISO/IEC 9796 に対する

[25] Diffie-Hellman 決定問題とは、G の元の組 (a, b, c) が与えられたときに、$(a, b, c) = (g^x, g^y, g^{xy})$ を満たす (x, y) が存在するかどうかを判定する問題である。(x, y) の存在のみが判定できれば良い（つまり、問題の (a, b, c) が Diffie-Hellman 鍵共有における公開鍵・中間鍵・共有鍵の性質を持つかどうか）。

Coron-Naccache-Stern 攻撃で明らかになったように、安全性証明を持たない公開鍵暗号では素因数分解問題や離散対数問題の困難性といった根本的な仮定に依らない思わぬ脆弱性が露呈する場合があり、安全性証明の価値がより一層評価されている。一方、現在の安全性証明で良く用いられるランダムオラクルモデルの妥当性もより研究されなければならないだろう。

また、今後、公開鍵暗号がより一層重要なインフラストラクチャー技術になっていくことを考えると、Shor のアルゴリズムが明らかにしたように、素因数分解問題や離散対数問題は量子計算機を用いれば多項式時間で求解されてしまうことは心に留めておかなければならないかもしれない（量子計算機が直ちに実現されることはあまり考えられないとはいえ）。

3.3 ハッシュ関数

3.3.1 ハッシュ関数とその要件

ハッシュ関数（Hash Function）は、パスワード認証・デジタル署名・メッセージ認証などの目的で幅広く用いられ、現代暗号の重要な分野を構成している。一般的には、ハッシュ関数とは、ある大きな領域 D からより小さな領域 R への多対一のマッピングを行う関数 h であり、

$$h:D\to R \quad \text{かつ} \quad |D|>|R|$$

$$|x| \quad \text{は領域の大きさ}$$

とかける。

特に、現代暗号におけるディジタル署名などではハッシュ関数としては、一方向性ハッシュ関数が重要となる。一方向性ハッシュ関数とは、関数値 y から $h(x)=y$ となるような元の値 x を求めることが「計算量的に困難な」ハッシュ関数である。ここで、関数 f が一方向性を持つとは、

［一方向性の定義］

以下の条件を満足するとき、関数 f を一方向性と呼ぶ。

｛f の容易性｝　x を入力して $f(x)$ を出力する多項式時間アルゴリズムが存在する。

｛f^{-1} の困難性｝　全ての（確率的）多項式時間アルゴリズム A に対して、以下が成立する。

$$\Pr\left[A(f(x))\in f^{-1}(f(x))\right]<\varepsilon$$

デジタル署名に必要な一方向性ハッシュ関数としては、理論的には、Naor と Yung により定義された汎用一方向性ハッシュ関数と Damgrad により定義された衝突困難ハッシュ関数がある。汎用一方向性ハッシュ関数とは、関数 h とその定義域内にある x が与えられたとき、$h(x)=h(z)$ となる z を求めることが困難であるような関数であり、衝突困難ハッシュ関数とは関数 h が与えられたとき $h(x)=h(z)$ となる一対の値 (x,z) を求めることが困難な関数といえる。

この２つのハッシュ関数は一定の仮定の下で、安全性が証明されている点で重要であるが、現実の世界ではハッシュ関数の出力である $y = h(x)$ は固定された有限の bit 長であることが必要であり、計算時間も有限の長さであることが要求されている。そこで、実際の「実用的」ハッシュ関数は非衝突一致性（Collision Resistance）を目標に設計されている。この非衝突一致性とは、

$$M \neq M' \text{かつ} h(M) = h(M')$$

なる関係を満たす異なるメッセージ M, M' を見つけることが困難であることと定義される。この性質は、正しいメッセージ M から偽のメッセージ M' を作り出すことが難しいことを意味している。

このような非衝突一致性をもつ実用的なハッシュ関数の構成法としては、

1. 既存のブロック暗号を連鎖（CBC）モード的に用いて実現する方法
2. 専用のハッシュ関数を独自に設計する方法
3. 法剰余演算を利用する方法

がある。

3.3.2 ハッシュ関数への攻撃

ハッシュ関数の安全性（非衝突一致性）の検証を行う上で、共通鍵暗号の安全性の検証と同様、「攻撃」の立場から検証を行う必要がある。

ハッシュ関数に対する攻撃法としては、(1)Birthday attack (2)Pseudo-collision and compression function attack (3)Chaining attack などの攻撃法がある。以下に、最も重要な攻撃法である Birthday attack の概略を示す。

Birthday attack は関数に依存しない攻撃法であり、この攻撃法に対する安全性は、ハッシュ関数の出力 $h(M)$ の長さに依存する。一般にハッシュ関数の出力の長さが、N bit であるとき、おおよそ $2^{N/2}$ 個程度の M' に対してハッシュ関数を計算することで、$h(M) = h(M')$ なる M' を見いだすことができることが知られている。この攻撃法の由来は、ランダムな23人が集まったとき、確率1/2で同一の誕生日の人間が2人以上存在するという直感に反する事実（Birthday attack）に由来している。

この Birthday attack に対処するためにハッシュ関数の出力 $h(M)$ はある程度長くなければならず、現在実用化されているハッシュ関数の出力 $h(M)$ の長さは、128 bit から196 bit 程度がふつうである。

3.3.3 ハッシュ関数の種類と用途

一般に、ハッシュ関数はその用途に応じて、Keyed Hash 関数と Unkeyed Hash 関数を使い分ける（図 3.3-1 参照）。改竄などの不正を検出する場合には、Unkeyed Hash 関数を使用するのが一般的であり、メッセージの作成者の確認などのメッセージ認証のような場合には、秘密通信の場合の共通鍵暗号と同様に、Keyed Hash 関数を使用するのが一般的である。

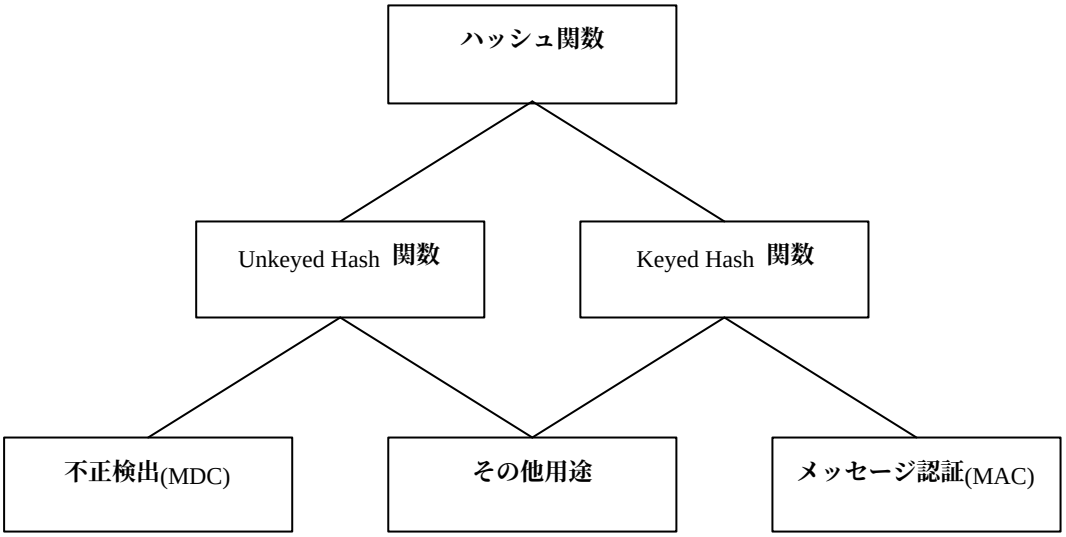


図 3.3-1　ハッシュ関数の種類と用途

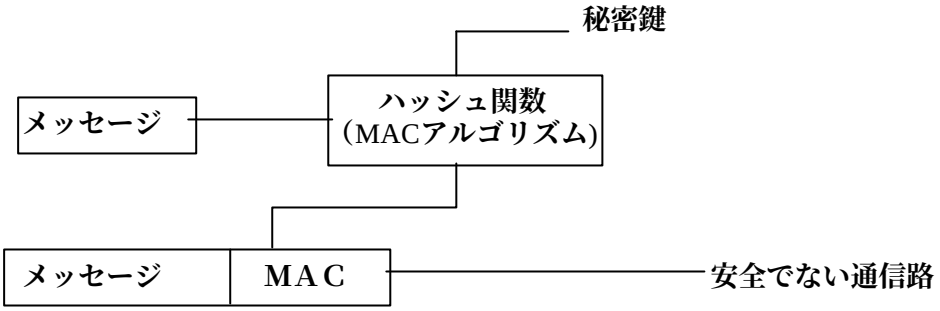
3.3.4　メッセージダイジェストの意味と利用法

メッセージダイジェストとは、大きな文章に電子署名を施すとき、ハッシュ関数を用い、元になる長いメッセージや文書、データファイルといった可変長の入力で受けたメッセージに対しハッシュ関数を用い、固定長の文字列で返す、コンピュータ処理手法のことである。

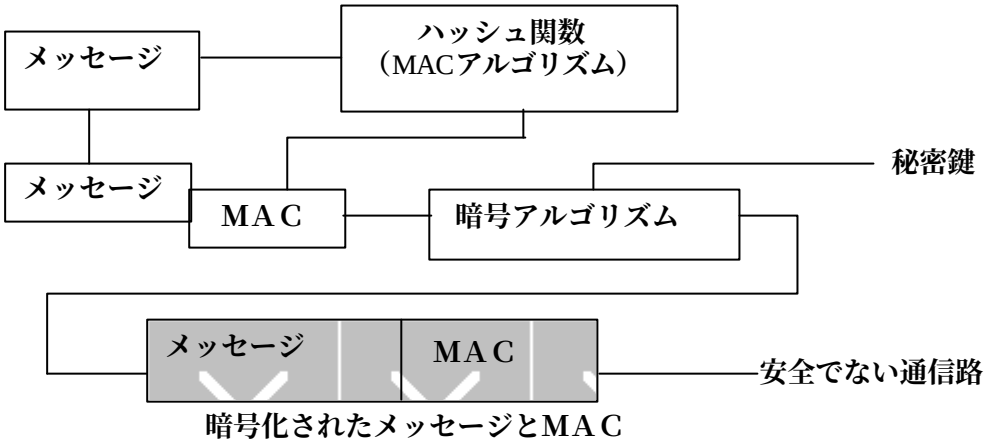
この固定長の文字列をメッセージダイジェストという。これは、長いメッセージや文書をコンピュータ処理することによって、簡潔なダイジェストとして表すという考え方で、元の入力情報の「デジタル指紋」ととらえることもできる。

メッセージダイジェストを用いる場合には、使用する通信路の安全性やメッセージ自体の重要性を考慮してどのような利用形態を選択するか決定する必要がある。その代表的な場合を図 3.3-2 に示す。図 3.3-2(a)はメッセージ自体は公になっても問題が少ないが、改竄されると問題が生じるような情報（例えば、電子商取引での発注数量など）を扱う場合の例であり、この場合にはメッセージ認証のためにハッシュ関数を利用している。図 3.3-2(b)は、メッセージ自身も公になると問題が生じるような情報（例えば、電子商取引でのクレジットカード番号など）を扱う場合であり、メッセージダイジェストを付加した後に暗号化を行い、送信する情報全体を保護している。図 3.3-2(c)は扱う情報は容量の関係で、安全でない通信路（例えば、インターネット）で送らざるを得ないが、通信路容量に制限があるが安全な通信路（例えば、ダイジェストのみをハンドキャリーする様な場合）の使用が可能な場合である。この場合には、メッセージ自体の保護は別として、メッセージダイジェストのみを安全な通信路で送る方式である。この場合には、Keyed Hash 関数を用いなくても十分に安全性が確保できる。

(a) MAC だけのデータ完全性保証方式



(b) 暗号化を併用したデータ完全性保証方式



(c) 安全な通信路を仮定したデータ完全性保証方式

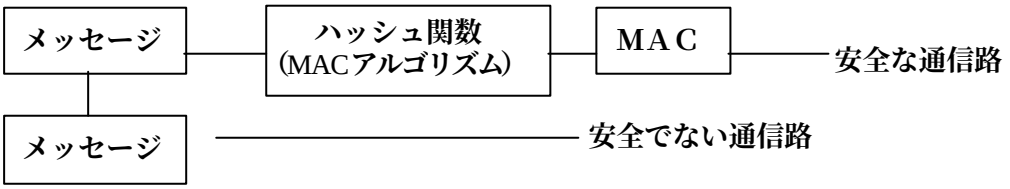


図3.3-2 メッセージダイジェストの種類と用途

3.3.5 各種メッセージダイジェスト用ハッシュ関数

(1) MD2, MD4, MD5

MD2, MD4, MD5 (MD はメッセージダイジェストの略）は広く使用されているハッシュ関数で、Ron Rivest が暗号化専用に設計したものである。これらは、128 bit のダイジェストで、徹底的に検索を行う以外に解読する方法はない。

処理速度は、32 bit CPU 上のソフトウェア実装では、MD2 が最も遅く、MD4 が最速である。MD5 は Rivest が「安全ベルト付き MD4」と称したもので、MD5 が MD4 よりも保守的な設計で安全性は高いものの、速度の面では MD4 よりも約33％遅くなっている。3つのアルゴリズムの中では MD5 が最もよく使用されている。MD4 と MD5 の使用に制限はなく、だれでも利用できる。MD2 は PEM とともに使用される。

MD2，MD4，MD5 の各詳細は、サンプルのCコードプログラムとともに RFC（Requests For Comments）の 1,319，1,320，1,321 でそれぞれ入手できる。

最近の理論研究で興味深い構造特性が分かってきているが、MD アルゴリズムに対する不正な侵入行為は発見されていない。

(2) SHA

SHA は、米国商務省に所属する標準化機関である NIST が規定した SHS（Secure Hash Standard）として定めたハッシュ関数で、米国政府の規格（federal information processing standard)として採用された。この SHA を改訂したハッシュ関数が SHA-1 である。SHA は、デジタル署名規格 DSS とともに使用するよう設計された標準である。SHA により、2^{64} bit 以下のさまざまな長さの入力から160 bit のハッシュ値が作成される。SHA の構造は MD4 や MD5 とよく似ているが、作成されるメッセージダイジェストは MD 関数のものより25 %長く、速度は約25 %遅い。安全性に関しては MD5 より高いと言われている。具体的なアルゴリズムの詳細は、参考文献[3]を参照されたい。

3.3.6 ブロック暗号を利用したハッシュ関数

暗号化鍵 K および（固定長）メッセージ M を持つブロック暗号を $B_K(M)$ とし、関数 H を

$$H(M) = B_K(M) \oplus M$$

と定義するとき、ブロック暗号 $B_K(M)$ が安全ならば、$H(M)$ は非衝突一致性を持つと考えられている。

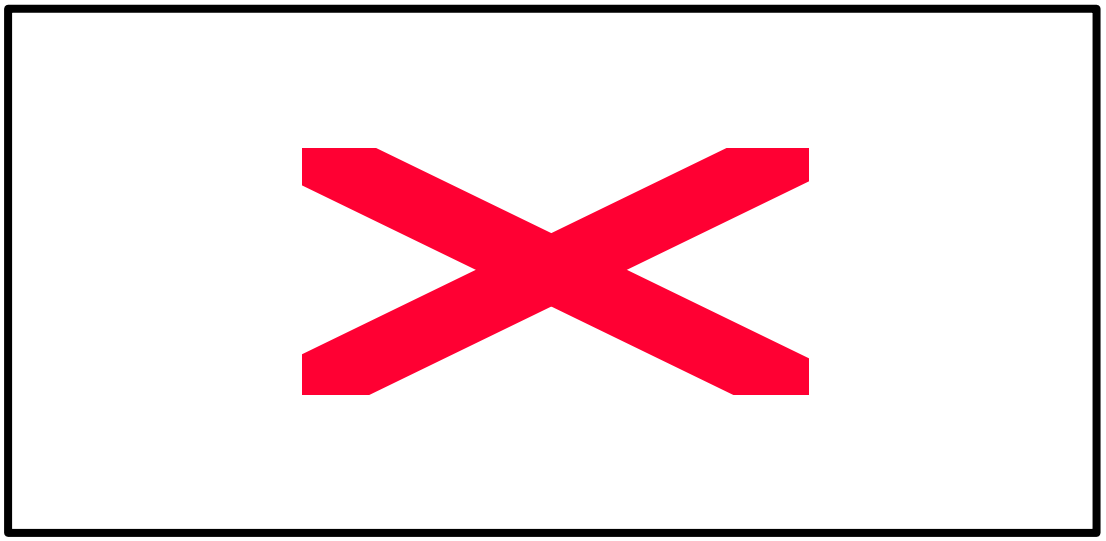


図3.3-3　ブロック暗号に基づくハッシュ関数の基本形

この例では、メッセージ M の長さは、基本となるブロック暗号 B が規定しているブロック長の整数倍の長さに限定される。そのため、可変長のメッセージに対しては、ブロック長の整数倍となるようにパディングを施す必要がある。この方式の特徴は、以下の通りである。

［長所１］：ハッシュ関数の安全性をブロック暗号の安全性の問題に帰着させることができる。

［長所２］：ブロック暗号の並列構造を利用し、処理速度の向上が可能となる。

［短所１］：ハッシュ値の長さと、ブロック暗号のブロック長が同一であるとハッシュ関数の強度が不足する場合が多く、２つ以上のブロック暗号を組み合わせることが必須である。

［短所２］：一般的に、総計算量が多く、高速化のためには、並列処理可能な環境が必要となる。

3.3.7 ハッシュ関数の特性と応用

ハッシュ関数の MAC 以外の応用として、

(1) 鍵の導出（鍵管理への応用）

POS 端末などで、現在のセッション鍵を露呈することなく、次のセッション鍵を生成する場合や、一方向性関数をベースにした「One-time password」の生成。

(2) 擬似乱数の生成

一方向性関数を使って、擬似乱数生成を行う場合には、発生される系列の性質が明らかにしておかねばならない。一般に暗号化関数を擬似乱数生成に使った場合、生成される系列は「暗号学的に良い乱数である」と保証することはできない。

これらの応用に用いる一方向性ハッシュ関数の持つべき性質は以下の通りである。

① non-correlation

② near-collision resistance

③ partial-preimage resistance（local one-wayness）

3.3.8 現時点での問題点

現在までに提案されているハッシュ関数の多くは、衝突が報告されており、用途によっては安全性に問題を生じる場合がある。従って、ハッシュ関数を用いる場合には、その関数の性質とその用途を考察し、リスクを考慮すべきである。このハッシュ関数応用に関する基準の明確化も課題として残されている。

3.3.9 今後の展望

我が国では、ハッシュ関数に関する研究がようやく始まったばかりであり、特に安全性の評価技術の確立が急務である。ハッシュ関数は、MAC やデータの完全性保証に多用されるため、今後のいっそうの研究が待たれる。

3.4 乱数

3.4.1 乱数とは

乱数は、サイコロを振る、ルーレットを回す、といった方法でゲームに偶然性を導入する道具として古くから使われている。その後、モンテカルロ法・オペレーションズリサーチ、サンプリング統計・品質管理などの道具として、その性質が論じられるようになった。近年になりセキュリティシステムにも欠かせない道具として注目されている。

乱数を発生させる方法には大別して二つあり、その一つは物理的手段で発生させる**物理乱数**であり、もう一つは算術式で発生させる**算術乱数**である。言い換えれば、**真性乱数**と**擬似乱数**である。

(1) 真性乱数

真性乱数の性質には、全くの不規則、平等な確率、前後が無関係、周期が無い（無限に続く）などがある。

物理乱数は、ランダムに変化する物理現象を利用して発生させるものであり、パワー・スペクトルが白色スペクトルであり自己相関係数がゼロとなるような物理現象を選択し、適切に数値化すれば一様乱数が得られる。

物理乱数は独立事象であり、棄却法（0-0:棄却, 0-1:0, 1-0:1, 1-1:棄却）やくくり合わせ法（0-0:0, 0-1:1, 1-0:1, 1-1:0）で性質を改善できる。

(2) 擬似乱数

算術乱数は一定の算術式で発生させたものであり、何らかの規則性が残り、真性乱数とはなり得ないので擬似乱数と呼ぶ。擬似乱数の発生法には、古くから使われているレーマー法のほかブロック暗号やストリーム暗号を利用したもの、カオス関数およびハッシュ関数を利用したものなどがある。

算術乱数には発生法に固有の規則性があり、目的と発生法に合わせて性質を改善する方法を考案する必要があるが、ソフトウェアのみで手軽に発生できることと再現性があるため広く使われている。クヌースの準数値算法／乱数にも、算法M、算法Bなど複数の乱数生成法と蓄積テーブルによる改善法が示されている。再現性は、モンテカルロ法への利用やシステム開発時のテストとしては好ましい性質であるが、セキュリティシステムの本番運用としては好ましくない。

モンテカルロ法などに使う場合と、暗号やチャレンジレスポンスといったセキュリティシステムへの応用では、算術乱数に対する要求事項に差がある。主なものとして、不可逆性と初期値のアドレス空間がある。生成した擬似乱数列から多項式時間で生成式と初期値が推定できないこと、過去に生成した擬似乱数列から以後生成する擬似乱数列が予測できないこと。初期値として 128bit 以上のアドレス空間がほしい。

セキュリティシステムへの利用では、初期値としてランダム要素を導入することで、発生数列の予測を困難にすることも可能である。しかし、ランダム要素が真性乱数何 bit 分換算の変動範囲かを検証した上でシステム設計する必要がある。

3.4.2 セキュリティシステムにおける乱数の役割

セキュリティシステムにおいては、システムの仕組みと運用方法を全て公開しても高いセキュリティレベルが達成できる事が望ましい。そういった要求を満たすには、暗号技術

による通信・保存、相手の認証など様々な局面で誰にも予測できない真性乱数の継続的供給が必要となる。

真性乱数の「全く規則性がない」、「全く統計的偏りがない」という性質がセキュリティシステムにおいて重要な位置を占める。真性乱数のそれらの性質を言い換えれば、いかなる方法でも理論的に（計算量的にでなく）予測不能であり全数探索の範囲を狭めるような工夫も不可能である。

主な用途として、各種パスワードの生成、暗号鍵の生成、ID 情報の生成、署名付加情報の生成などがあり、暗号技術、ハッシュ技術と並ぶ重要な技術である。良質な真性乱数から直接生成した暗号鍵であれば、実用上他の鍵との重複を配慮する必要はない。

3.4.3 乱数の種類と性質

(1) 乱数サイコロによる乱数（物理乱数）

極めて安価に、かつ手軽に採用できる方法である。人間によるオペレーションが必要であり、高速・大量の乱数を発生させることはできないがパスワードの類には適している（図 3.4-1 参照）。

乱数サイコロ無しでパスワードの設定を要求すると、氏名・生年月日・電話番号・住所の一部・口座番号・日付・時刻・適当な単語などが採用される事が多い。その結果として、辞書アタックといった最も一般的なハッキング手段が有効となる。パスワードの構成として英字・数字・特殊記号の混在を義務付けるなどの手法もあるが根本的な対策にはならない。

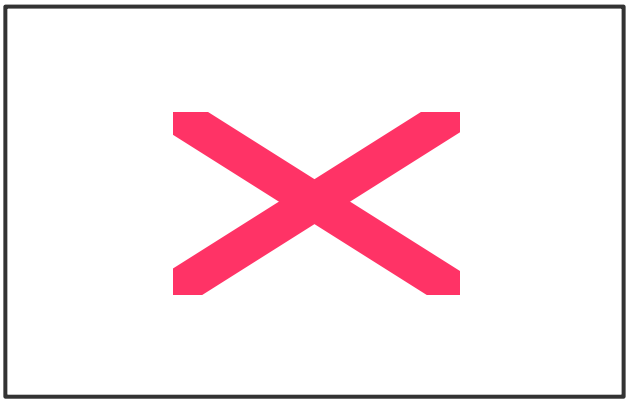


図 3.4-1　乱数サイコロ

(2) 抵抗体の熱雑音による乱数（物理乱数）

抵抗体の熱雑音は、非常に多数の電子が作る微少電流を合成したものであり、それぞれの電子の衝突・錯乱は相互に無関係に発生しているため、確率密度関数はガウス型になる。すなわち、一様乱数を生成することになる。

高抵抗の両端に発生する微少な雑音電圧を、ハイインピーダンス・低雑音のアンプで増幅すれば使いやすい雑音電圧を得られる。その電圧を A/D コンバータでデジタルデータとすることで乱数列が得られる。

いったんデジタルデータになれば、乱数のある程度の検定および性質改善は容易である。棄却法、くくり合わせ法、変換誤差と電圧の不安定さを避けるための下位 bit と上位 bit の切り捨て、デジタル信号処理による周波数シフトなど低域成分の切り捨て、後述の Mersenne Twister 法など良質な擬似乱数との排他的論理和など様々な方法が考えられる。

この方法の利点は、自動的にかつ大量の乱数を高速生成できることであり、パスワードへの応用ではアルゴリズムが漏洩すれば無力な一定の数式で発生させる方式と違い、完全なワンタイムパスワードが実現できる。

最近、Intel のチップセット 810 を皮切りに Intel(R) Random Number Generator を内蔵を始めた。また、東芝からも PCI バスのボードとしてランダムマスターが商品化されている。

(3) キー入力のコードおよび入力時間やマウス移動の軌跡および移動速度のバラツキによる乱数（ランダム要素）

ランダムにキー入力やマウスを移動させて、そのコードとキー入力時間や軌跡とマウス移動時間のバラツキを計測することでランダム性のある数値を得られる。

この方法の利点は、特別なハードウェアが要らないことであるが、乱数としての性質が悪く規則性があること、人間によるオペレーションが必要なことが欠点である。また、キーボードプロセッサのクロックやマウスの検出クロック以上の分解能は得られない。

オペレーションについては、普段からキー入力時間等を計測してその時間のバラツキを蓄積しておくことで回避できるが、規則性はさらに強くなる。したがって、他の要素と組み合わせて補助的情報として使うのが良いと思われる。

(4) FD，HD など媒体へのアクセス時間のバラツキによる乱数（ランダム要素）

FD，HD などのアクセス時間のバラツキを計測することでランダム性のある数値を得られる。

この方法の利点は、特別なハードウェアが要らないことの他、人間によるオペレーションが不要なことがあげられるが、乱数としての性質が悪く規則性があることが欠点である。CPU 実行命令のシーケンスにより確定的に決まるので原則として処理するデータや事象によるバラツキしかない。

したがって、この方法も他の要素と組み合わせて補助的情報として使うのが良いと思われる。

(5) リアルタイムクロックの値による乱数（ランダム要素）

ある瞬間におけるリアルタイムクロックのカウンタ末尾桁には、ランダム性があるとみなせる。

この方法の利点と欠点は、前記 FD，HD など媒体へのアクセス時間によるものと同様である。したがって、この方法も他の要素と組み合わせて補助的情報として使うのが良いと思われる。

(6) レーマー法／線形合同法（算術乱数）

古くから多用されている乗算合同法および混合合同法であり、長い周期と比較的良好な数列が得られる反面、生成された擬似乱数列を座標とする点を多次元の単位空間に打点すると、少数個の平行超平面の上にすべての点が乗るという性質がある。他に LESR（線形フィードバックシフトレジスタ）法もあるが基本的には同様なので省略する。

乗算合同法　$x_{i+1} \equiv a x_i \bmod p$　　経験的に a は $\sqrt{p}$ より大きく $p-\sqrt{p}$ より小

さい値がよい

p は素数

混合合同法　$x_{i+1} \equiv ax_i + b \bmod p$　　b は、$0.211p$ に近く $4k+1$（k は整数）を満たす値がよい

(7) 平方採中法（算術乱数）

レーマー法より拡散度が高い反面、生成される擬似乱数列が可能な数全体の幾つかのグループに分かれ、各グループには１つのエンドポイントないしエンドループ（比較的短い周期）があるという性質と、ゼロの並びが拡大するという縮退現象があり、周期や系列相関が理論的に計算できないため単独で使われることはない。

＜例＞　$123{,}456^2$ → 15,241,383,936
$241{,}383^2$ → 58,265,752,689
$265{,}752^2$ → 70,624,125,504
$624{,}125^2$ → 389,532,015,625

(8) 暗号技術による乱数（算術乱数）

ブロック暗号に対して運用モードとして CFB モードや OFB モードを適用して擬似乱数列を得るほか、ストリーム暗号をそのまま使って擬似乱数列を得る。その擬似乱数列の性質に関する研究発表例は少ない。

(9) Mersenne Twister 法（算術乱数）

$2^{19937}-1$というメルセンヌ素数の素数性を使い、19937 次元のベクトル空間における固有多項式の既約性を判定することで周期が$2^{19937}-1$(約10^{6000})で、623 次元超立方体の中に一様に分布する（ように見える）擬似乱数列を高速生成する。初期値のアドレス空間が 2^{32} と狭いのと非可逆的アルゴリズムでない（通常の線形合同法）など、このままでは暗号および認証用の乱数としては使えないので、他の方法と組み合せて使うことが考えられる。

3.5 電子透かし

3.5.1 電子透かし技術の概要

電子透かし（Digital Watermarking）技術とは、「音声や画像などのディジタルコンテンツ内に、人間に知覚できないように別の情報（透かし情報）を付加・多重化する技術」である[1][2]。図 3.5-1 に、電子透かしの概念図を示す。一般に透かし情報は、コンテンツ全体に分散して埋め込まれている。また、コンテンツの複雑な部分に特に多く埋め込むことにより、人間の目（画像の場合）または耳（音楽の場合）では判別しにくくしている。また、ディジタルコピー直後は当然のことながら、一定の編集・加工を施した後でも透かし情報が残る特徴がある。そのため、著作権に関する情報を電子透かしとして埋め込むことにより、ディジタルコンテンツ流通時の著作権保護手段として用いることができる。コンテンツヘッダで著作権情報を管理する方法と異なり、電子透かしは、コンテンツと著作権等情報が一体化されているため、コンテンツの品質を劣化させることなく著作権情報だ

けを除去／改竄することは極めて困難である。電子透かしは、それ自体では、ディジタルコンテンツの bit 単位でのコピーを防止する機能はないが、複製しても透かしが維持され、証拠が保存されるところが紙幣の透かしとは性質を異にする。

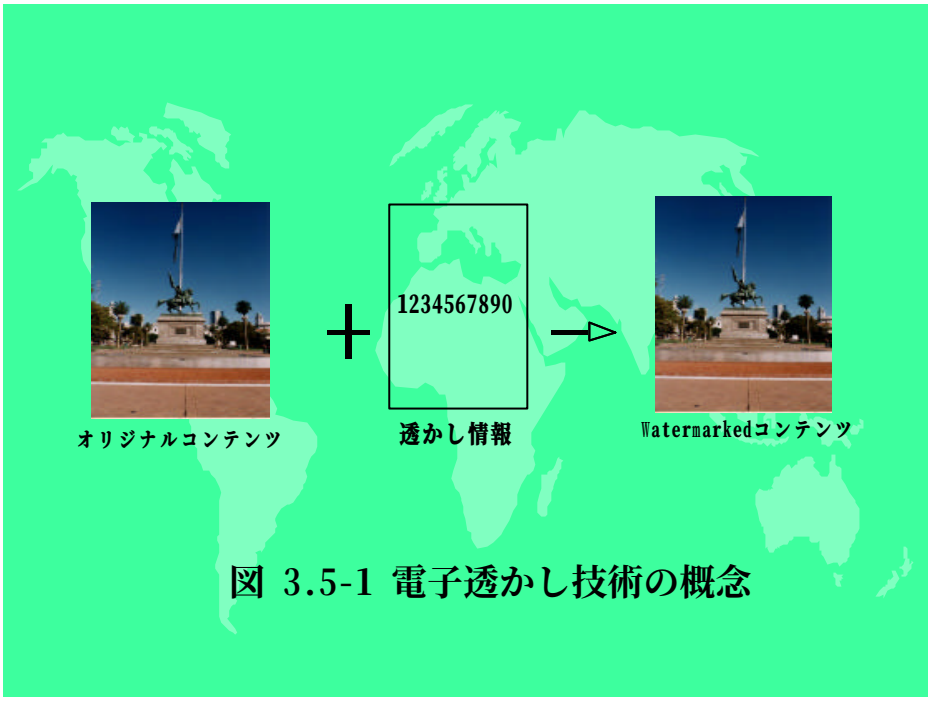


図 3.5-1 電子透かし技術の概念

3.5.2 電子透かしの演算

図 3.5-2 に、電子透かしの埋め込み、読み出し処理の一般演算式を示す。埋め込み演算処理は、オリジナルコンテンツ（C）と埋め込みたい透かし情報（S）及び、必要により鍵情報（Key）を入力して埋め込み済みコンテンツ（Watermarked コンテンツ（C_W））を生成する。読み出し演算処理は、Watermarked コンテンツ、必要によりオリジナルコンテンツ及び鍵情報を入力して埋め込まれている透かし情報を出力する。

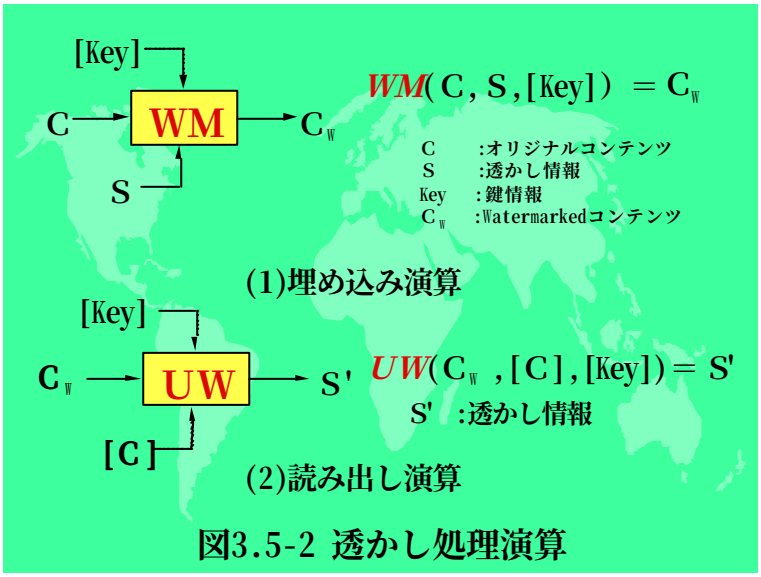


図3.5-2 透かし処理演算

3.5.3 電子透かし方式の分類

電子透かし方式を種々の観点から分類する。

(1) 透かし埋め込み空間

電子透かしの埋め込み空間で分類すると、大きく２通りの方式が提案されている。音声波形や画像の画素などコンテンツ標本値に直接埋め込む方式と、一旦、コンテンツを周波数変換(DCT，FFT，wavelet 変換など）した領域に埋め込む方式である（図 3.5-3）。一般に、標本値埋め込み方式は、処理速度は速いが、加工・圧縮などにより透かし情報が消滅しやすい性質がある。一方、周波数領域埋め込み方式は、処理が重いが、加工・圧縮耐性に優れている。どちらの方式も、いかに人間の目や耳では知覚できない領域／成分に埋め込むかが勝負となっている。

各メディアの特徴を活かした方式も有効である。静止画の場合、輝度が高い画素の周辺にある輝度の低い画素は見えにくいという視覚的マスキング効果を利用して、人間の目をごまかして埋め込む方法が提案されている。動画の場合、フレーム内において静止画の電子透かし方式を基本的に踏襲するのに加え、フレーム間において動画符号化の際の動きベクトルに透かしを埋め込む方式も提案されており、両者を効果的に組み合せて耐性向上を図っている。音楽の場合、静止画と同様、聴覚的マスキング効果（大きな音に重畳された小さい音は聞こえにくい）を利用して埋め込む方式や、残響効果（実際の音が消えても耳には音が残っている）を狙って短い時間差でかけたエコーに透かしを重畳する方法などが提案されている。

図3.5-3 電子透かし埋め込み空間

(2) 読み出し時オリジナルコンテンツの要／不要

埋め込み情報の読み出し方法として、埋め込み済みコンテンツのみから行う方法と、オリジナルコンテンツと比較して行う方法がある（図 3.5-4）。前者はオリジナルコンテンツが不要なため、適用範囲が広い。例えば、利用者がインターネット上で流通するコンテンツから透かし情報として著作者 ID などを読む場合、オリジナルコンテンツの入手は困難で、埋め込み済みコンテンツのみから読み出す必要がある。ただし、不

正コピーなどにより流通しているコンテンツの特定を行う際には、著作権者または代行者が行うことが多いため、高い読み出し精度が期待できるオリジナルコンテンツとの比較による読み出し方法が効果的である。

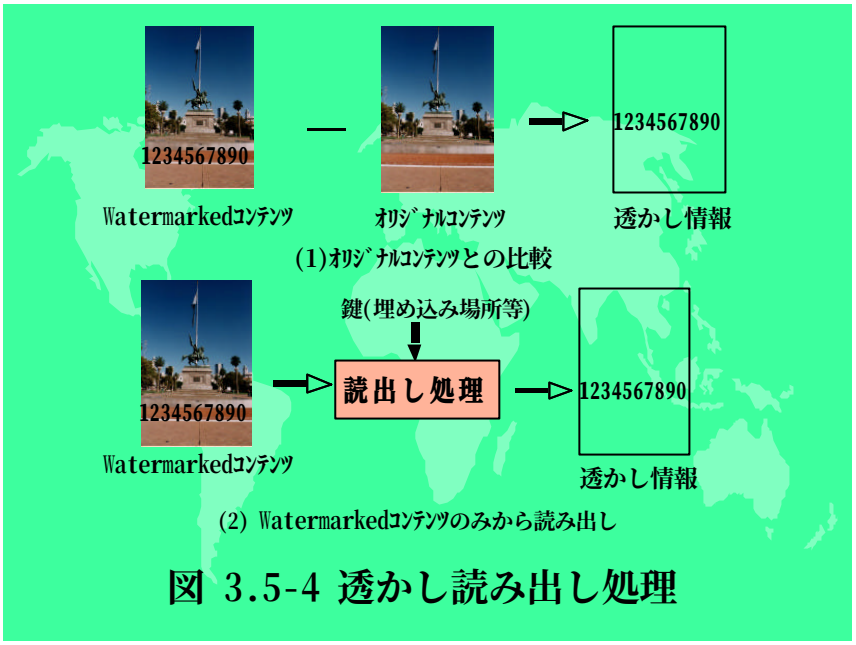


図 3.5-4 透かし読み出し処理

(3) 見える透かし／見えない透かし

電子透かしは、人間の目／耳で知覚できないのが一般的であるが、見える透かし方式もある。その目的は、見える透かしとすることにより、明示的に著作権主張及び警告を行う場合や、見本であることを示す場合に用いられる。なお、購入処理など正規の所有権手続きを完了すると見える透かしが消えたり、見えない透かしに変わることも可能である。

3.5.4 電子透かしの応用

(1) ディジタルコピー制御

DVD などのデジタル記録媒体に記録されたデジタル動画、音楽に電子透かしでコピー制御情報（数 bit 程度の透かし情報で「１回だけコピー可能」「コピー禁止」「コピー自由」の状態を表現）を埋め込んでおくことにより、以降のディジタルコピー操作の際に、デジタル機器がそのコピー制御情報に従った動作を行う（図 3.5-5）。デジタル記録媒体とその再生装置のハードウェア、ソフトウェアが協調して不正コピーを防止する。

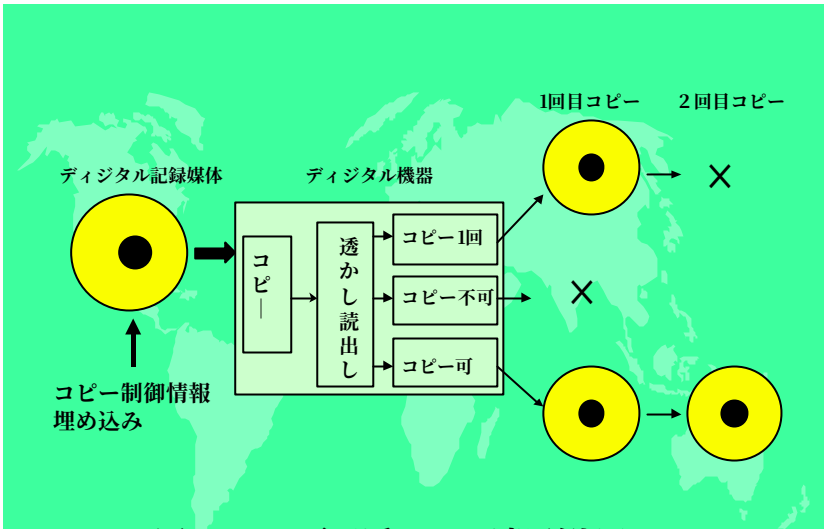


図 3.5-5 電子透かしの適用例(1)

(2) 著作権処理の利便化

透かし情報として、著作権者の関連情報を埋め込むことにより、デジタル・コンテンツの権利者や使用条件を明示することができる。インターネットなどで流通しているデジタル・コンテンツを利用・購入したい場合、電子透かしにより著作権情報を読み出すことにより権利処理手続きが容易となる（図 3.5-6）。本適用例では、埋め込み処理は著作権者側で行い、読み出し処理は利用者側で行う。利用者側で電子透かし情報を読み出すため、埋め込み場所を特定している鍵情報を利用者が容易に判別できないように、読み出しプログラム内で管理することが重要である。米 Digimarc 社では、「著作権管理サービス - Marc Center -」と称して静止画に対する著作権情報提供サービスを行っている[1]。

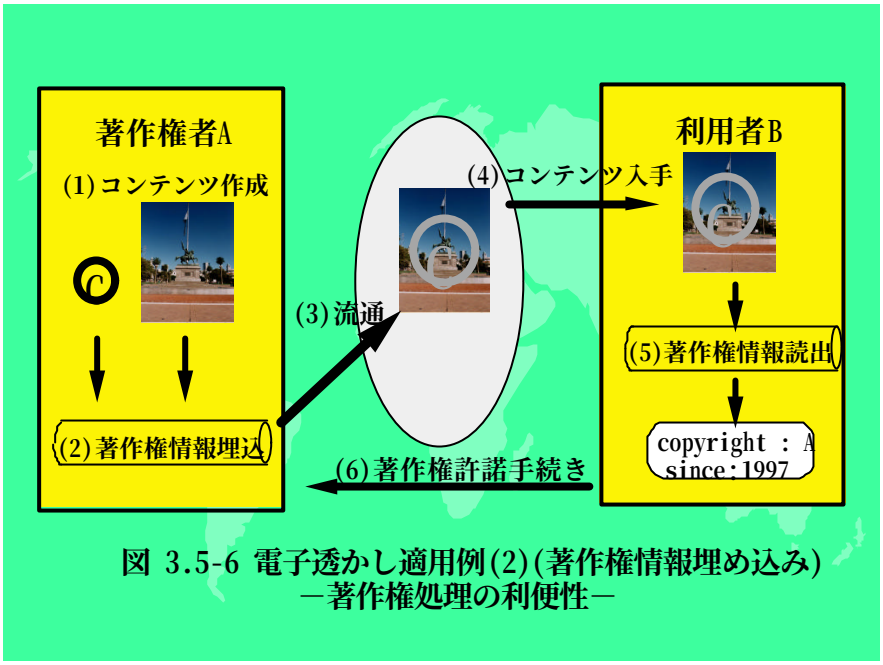


図 3.5-6 電子透かし適用例(2)(著作権情報埋め込み)
－著作権処理の利便性－

(3) 不正コピーの抑止

透かし情報として、デジタル・コンテンツを購入した利用者の情報を埋め込むことにより、販売先や、利用者が不正にコピーして海賊版を自分のホームページで展示・販売などをすることを抑止する適用例である（図 3.5-7）。本適用例では、埋め込み／読み出し処理は、共に著作権者側で行う。そのため、鍵情報の管理は前記(2)の適用例に比べれば容易である。

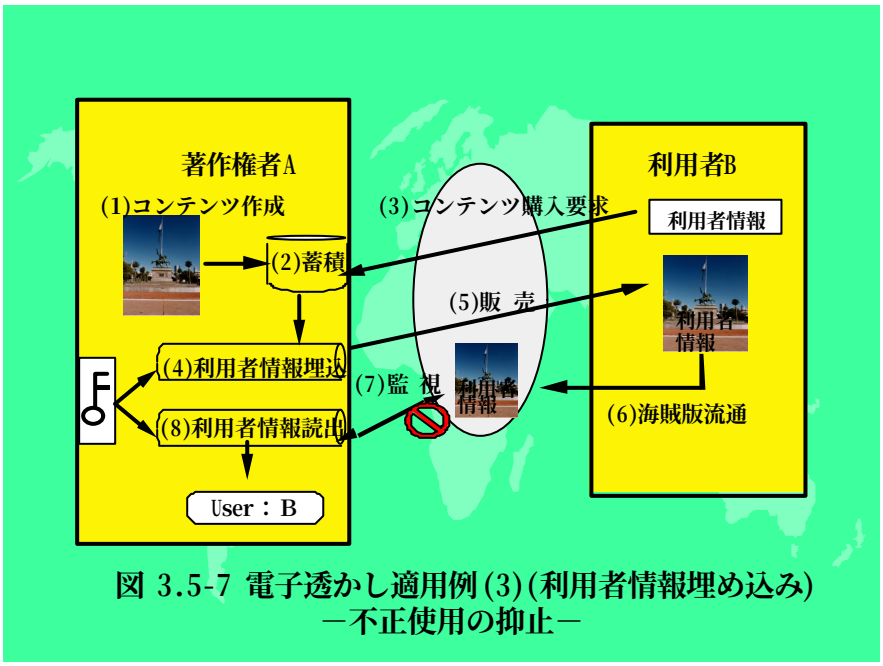


図 3.5-7 電子透かし適用例(3)(利用者情報埋め込み)
－不正使用の抑止－

(4) 真正性の検証

電子透かし情報とデジタル署名技術を組合せることにより、証明写真などが改竄されていないことを検証できる[3]。例えば、デジタルカメラなどで建築工事現場や交通事故状況を撮影する際、自動的にカメラ内で、撮影日時などを電子透かし情報で写真内に埋め込むと同時に、その写真データのデジタル署名を計算しておく。写真を提出する際、そのデジタル署名データも合わせて提出することにより、検証者は署名データを検証し電子透かしを読み出して、撮影日時とその真正性を同時にチェックすることが可能となる。

3.5.5 標準化の動向

電子透かしの各種標準化の動向をまとめる。

(1) CPTWG (Copy Protection Technical Working Group)

CPTWG は、家電メーカー、コンピュータメーカ、映画メーカーで構成された業界団体で、DVD に記録された映画などのコンテンツを DVD-RAM などのデジタル記録媒体へコピーする際の制御方式について、図 3.5-5 に示した通り電子透かしを用いることが検討されている[4]。

(2) SDMI (Secure Digital Music Initiative)

SDMI は、米国レコード協会 RIAA を中心にやオーディオ機器メーカーなどで構成され、インターネットでの音楽配送、次世代オーディオ機器などでの著作権保護の統一仕様を策定している。音楽コンテンツにコピー制御情報や著作権情報を記述する手段として、電子透かし方式が検討されている。

(3) CIDF (Content ID Forum)

CIDF は、通信、家電メーカーおよび、コンテンツ保有企業などが集まって結成された業界団体で、デジタル・コンテンツにユニークな ID（コンテンツ ID）を付与することにより、コンテンツの属性検索、利用履歴の探索などを一元的に出来るようにしようとすることを狙っている。CIDF の中で、電子透かしによりコンテンツ ID を埋め込むため、透かし方式の共通仕様の策定が行われている[5]。

3.5.6 電子透かしの課題

電子透かし技術を、デジタル・コンテンツの著作権保護手段として使用するための各種課題について述べる。

(1) 技術的要件

技術課題として、下記の要件を満たす必要がある。

- オリジナル情報を乱さない（コンテンツ品質維持）
- 透かし入りコンテンツの編集、加工、圧縮後も透かし情報が残り続ける（編集耐性）
- 透かし情報の意図的除去／改竄が困難（攻撃耐性）

(2) 共通評価基準の設定

著作権者やコンテンツ提供業者にとって、どの電子透かしを採用したら当該サービスの目的に最適かを判断するため、共通評価基準を以下の３つの観点から策定する必要がある。

- 編集耐性評価基準：
  コンテンツの切り貼り、変形、拡大／縮小、非可逆圧縮／伸張、印刷など編集・加工・圧縮等処理の標準セットを決定し、それらの処理後に電子透かし情報が残るか否かの処理耐性を評 価する。なお、その際、静止画、動画、音声などメディァ毎に評価対象となる「標準コンテンツ」の選定も必要となる。
- 品質評価基準：
  電子透かし情報を埋め込んだ場合、人間が知覚できない程度ではあるが、コンテンツは変化している。そのためにも客観的な品質評価尺度の導入が必要である。
- 攻撃耐性評価基準：
  前記の編集・加工・圧縮等処理以外での、各種の意図的な電子透かし解読・除去・改竄攻撃に対して、透かし情報が残るか否かの耐性を評価する。その際、コンテンツの価値を低下させる程度までの品質劣化を伴う透かし情報の解読・除去・改竄は「攻撃失敗」とみなす。

なお、編集耐性、品質評価基準については、静止画を対象として、評価基準が策定されている[6]。また、代表的な評価ツールとして、Petitcolas らにより開発された StirMark がある[7]。幾何学的変形、復号・符号化の繰り返しなどにより、静止画対応の各種電子透かし方式の耐性を評価できる。

(3) 電子透かしの証拠能力

不正コピーの摘発に際して、裁判所などの公的機関での抗争時、電子透かし情報が証拠としての能力があることを保証することが必要である。そのために、必要に応じて関連法規の整備・改正や今後の判例の積み重ねを期待する。

4 暗号鍵の管理

4.1 鍵のライフサイクル

4.1.1 作業鍵とマスタ鍵

通常、長いデータを暗号化・復号する場合には高速性を活かして共通鍵暗号アルゴリズムを用いるが、常に同一値の鍵を用いると解読されやすいので暗号化を多段に行うことが多い。そのときに用いられる鍵を作業鍵（あるいはセション鍵）と呼ぶことがあり、一般的にはその時に生成した乱数（実際には擬似乱数）データを用いる。また、作業鍵は 4.2 に述べるように公開鍵暗号や、共通鍵を生成させる鍵管理アルゴリズムを用いて暗号化して相手に配送する。この場合、作業鍵を暗号化する鍵をマスター鍵と呼ぶこともある。図 4.1-１に作業鍵の図を示す。

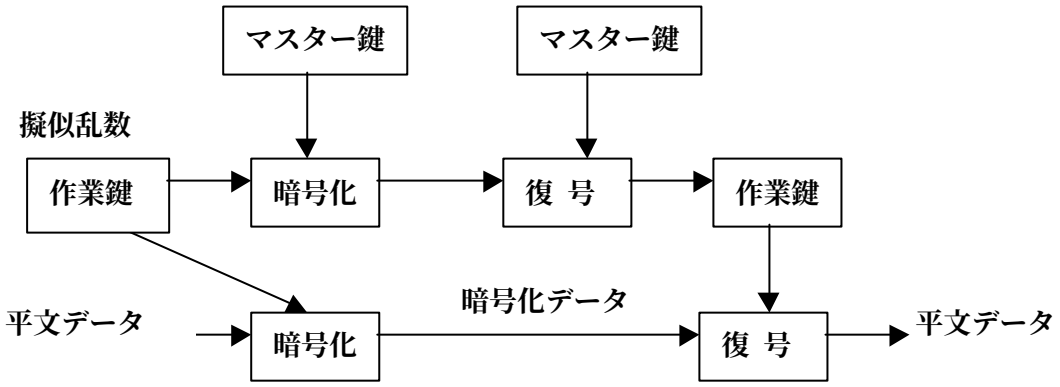


図 4.1-１　作業鍵の概念

4.1.2 鍵と証明書の有効期限

公開鍵暗号では、一旦生成した鍵を比較的長期間利用することになるため、攻撃者による長期間にわたる暗号解読に対抗すべく、鍵のサイズと有効期限を適切に設定し、鍵を定期的に更新する必要がある。有効期限は、予想される解読時間（3.1.4 参照）よりも短く設定し、有効期限内に解読される危険性を極力小さくする。

4.1.3 鍵運用の概要

秘密鍵が固定された簡単なシステムを除いて、安全性を高めるために暗号に用いられる鍵は定期的に更新される。

図 4.1-2 に、鍵の生成から廃棄までのライフサイクル示す。

(1) ユーザ登録（User Registration）

エンティティ（企業・個人を含む使用者）はこの登録により認証される。ここではパスワードやセキュアな PIN（個人識別子）、ワンタイム技術などを用い、初期証明書の取得が行われる。

(2) ユーザ初期化（User Initialization）

エンティティはユーザ登録時に得た初期証明書の使用やインストレーションにより

暗号アプリケーションを初期化する。

(3) 鍵生成（Key Generation）

暗号化鍵生成はエンティティが自身の鍵を生成するか、信用された機関から鍵を得ることが望ましい。

(4) 鍵インストール（Key Installation）

鍵は使用するユーザによりエンティティのソフト・ハードウェア内に、パスワードまたは PIN の手動登録、ディスクの配布、書込専用デバイス、IC カードまたは他のハードウェアトークンかデバイス（例えばキーローダー）、等の技術によりインストールされる。

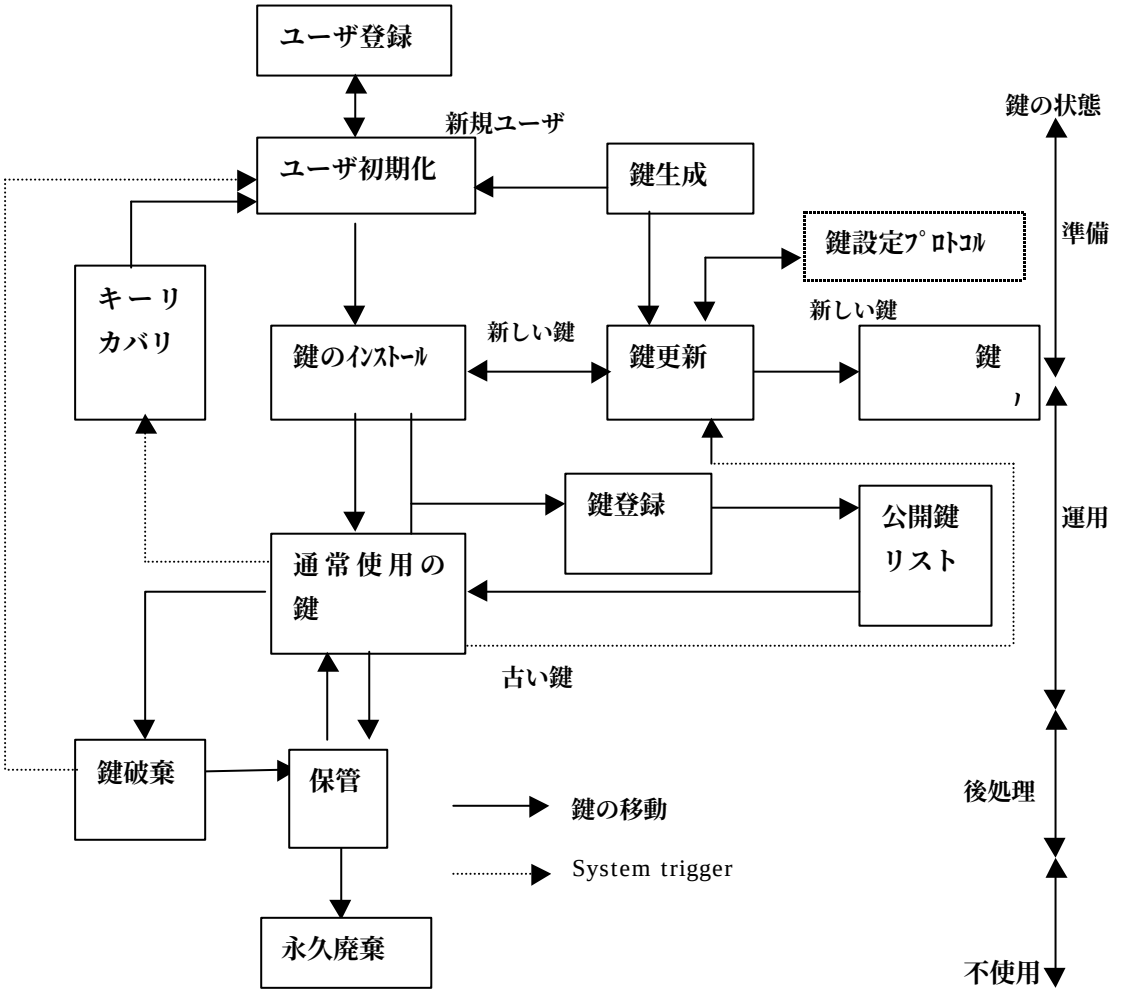


図 4.1-2　鍵のライフサイクル

① 鍵登録（Key Registration）

鍵インストールに関して、証明書はエンティティを識別する名前として公式に記録される。

公開鍵方式の場合、公開鍵証明書は認証機関で作られる。

② 通常使用（Normal Use）

ライフサイクルの目標は暗号の使用に対して鍵の運用操作を容易にすることであり、この状態は暗号使用期限まで続けられる。また公開鍵は暗号化に使用しなくなったとしても、秘密鍵は復号に（通常）使うために残される。

③ 鍵バックアップ（Key Backup）

鍵をバックアップするセキュアな保管メディアは、キーリカバリのためのデータソースを供給することが望ましく、またバックアップは運用使用のために短期間保管となる。

④ 鍵更新（Key Update）

暗号使用期限が終了すると、鍵は新たなものに置き換えられる。

⑤ 保管（archival）

使用しない鍵は、否認を含む論争の解決などで必要となる鍵回復のために、特別な環境の元で保存される。

⑥ 鍵登録解除と鍵消去（Key De-registration and destruction）

エンティティに関連した鍵や証明書の要求が長期間なかった場合、鍵に関するすべての公開記録から鍵登録は解除され、すべての鍵のコピーは消去される。

⑦ キーリカバリ（Key Recovery）

鍵を紛失した場合に、バックアップコピーから鍵を復元できる。

⑧ 鍵破棄（Key Revocation）

鍵の漏洩時にはその鍵を破棄する。

4.1.4 鍵の更新

(1) 作業鍵の更新

作業鍵とは、文字どおり対象データを暗号化するだけに利用される鍵であり、通常は共通鍵暗号が利用され、１回限りもしくは一定限度内（回数、期間）で更新されることが前提である。

新たな鍵に更新する際には、鍵の生成や、共有のための手続き（配送や共通鍵の生成）が必要となる。鍵の生成においては、万一それまでの鍵が解読された場合でも新たな鍵との関連性が無いように、十分性質の良い乱数を用いる必要がある（3.4 参照）。もちろん、暗号アルゴリズム毎に理論的に強度の低い鍵は使用を避けるべきである。鍵の配送と共通鍵の生成に関しては 4.2 を参照のこと。

なお、鍵の配送手段としては通信を用いるほか、人手を介して配送することが考えられる。この場合は通信よりも安全が保ちやすいが、以下のような点などで人間のセキュリティレベルに注意が必要である。

1. ８文字程度の短い鍵は電話連絡で済ますことも可能であるが、盗聴や、声を聞かれてしまう可能性がある
2. 文書による受け渡しでは、配送中の紛失や使用後の廃棄を厳重にしなければならない
3. 電子メールなど、盗聴されやすいメディアは避けるべきである
4. フロッピィや磁気テープなどの可搬媒体を使用する場合は、途中で複写される可能性があるので直接の手渡しが必要である
5. 同様に使用後の可搬媒体はデータを消去の上物理的に読めないようにデータの

上書きなどを行う

また、配送時の安全性を高めるために、鍵の階層構造を取ることがある。すなわち、平文を暗号化するための作業鍵を、上位の暗号鍵１で暗号化し、更にその上位の鍵で暗号化し、・・・ということを繰り返し、最終的にはマスター鍵で暗号化する。

この方式の優位点は、

① マスター鍵以外に共通鍵方式を用いる場合は、公開鍵方式による交換に比べ受け渡しするデータ量が少なくて済む

公開鍵：数百 bit（数十 byte）、共通鍵：～百数十 bit

② 共通鍵を n 回まで利用できるのであれば、作業鍵は n 回毎に更新が必要だが暗号化鍵 1 は n^2 回毎、・・・というように交換回数は少なくて良いといった点である。

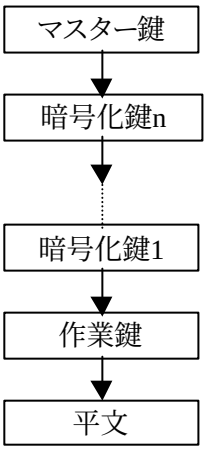


図 4.1- 3　鍵の階層構造

(2) マスター鍵の更新

マスター鍵も、作業鍵と同様にある一定限度内での使用にとどめ、期限がくれば更新しなければならない。認証機関のルート鍵も例外ではない。一般的には、共通鍵暗号によるマスター鍵は利用一定回数以内で更新が必要である。また、公開鍵暗号の場合には、鍵の bit 長と、その鍵の利用目的により認証機関側で有効期限を設定する。たとえば、エンドユーザの個人鍵は一般には２年程度である。

マスター鍵の更新時の配送は、上位の鍵による方法が不可能なため、主に物理的な手段が用いられるので、上記(1)のような点に注意を払う必要がある。仮に、それまで使用していた鍵を使って暗号化して配送してしまうと、危険度の高くなった古い鍵がいずれ解読された場合、その鍵を使って次の鍵を求め、・・・と直ちに現在使用中のマスター鍵が求められてしまうからである。マスター鍵は通常、鍵管理システムで廃棄鍵として保管される。公開鍵暗号方式での廃棄鍵の運用は 4.1.5 に述べる。

4.1.5 鍵の廃棄

(1) 鍵の寿命と廃棄

鍵の有効期限内において、鍵の紛失や他人への漏洩が認められた場合や、鍵の所有者の移動などにより鍵が不要になった場合に、それらの鍵を廃棄し、それらの鍵が使われても無効であることを全ユーザに周知するしくみが必要である。このための手段としては、例えば X.509[1]で規定された CRL（Certificate Revocation List：証明書失効リスト）と呼ばれる廃棄鍵のリストを利用する。

(2) 廃棄鍵のリストの管理と運用

公開鍵暗号を利用する場合、鍵が有効なものかどうかを知ることは非常に重要であり、全利用者が、リアルタイムに廃棄鍵のリストを参照できることが理想である。しかし、現実には、利用者の多い大規模ネットワーク環境での利用や、IC カードなどオフラインで鍵が利用される場合など、廃棄鍵のリストを管理・運用する上での課題は多い。以下に、廃棄鍵のリストの管理運用上の課題について列挙する。

1. 大規模ネットワーク環境での利用においては、廃棄鍵のリストを１個所で集中管理することは、リストが膨大となり、負荷が集中して非効率的である。
2. 廃棄鍵のリストを分散管理する場合、各廃棄鍵のリストを同期を取って更新する必要がある。
3. オフライン環境での利用においては、廃棄鍵のリストを参照するしくみが必要である。
4. 廃棄鍵のリストの運用管理には相応のコストがかかる。利用者の規模、対象データの重要度、ネットワーク環境など、局面に応じた運用管理を行う必要がある。

4.2 鍵の配送

4.2.1 鍵の配送方法と配送アルゴリズム

共通鍵暗号は、データの暗号化と復号とに同じ鍵を用いることから、この鍵を受信者にどのように知らせるかが問題となる。第３者に知られないように鍵を共有する必要があり大規模システムでの鍵配送は非常に手間のかかるものとなる。

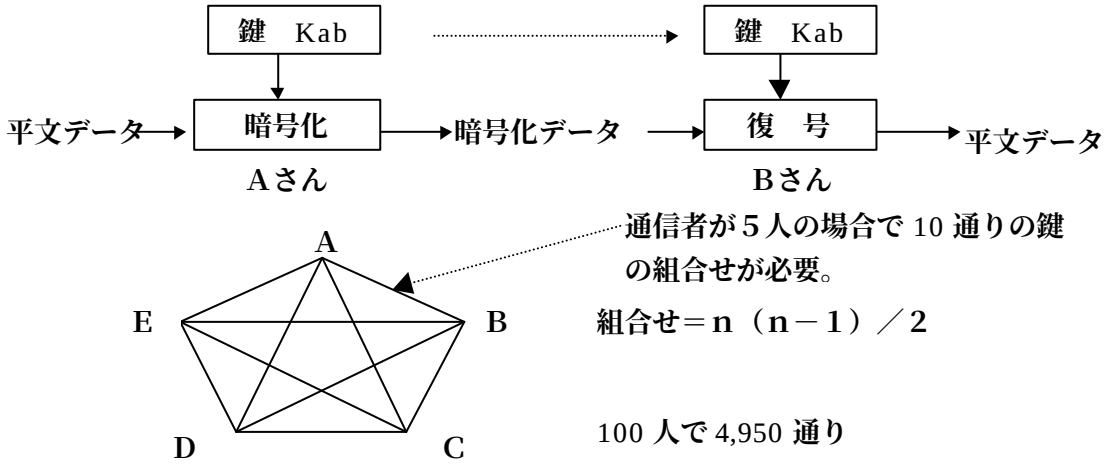


図 4.2-1　鍵の配送

このため、次に挙げるような鍵配送アルゴリズムが開発され実用に供されている。

1. 共通鍵の事前配送
2. Kerberos（時刻印を用いた共通鍵方式）による鍵配送
3. RSA による暗号化鍵共有
4. ElGamal による暗号化鍵共有
5. Diffie-Hellman による共通鍵の生成
6. KPS(Key Predistribution System、1986 年に松本・今井によって開発された暗号鍵共有方式)による共通鍵の生成
7. その他　楕円暗号、MTI（松本、高島、今井）

4.2.2 共通鍵暗号による暗号鍵共有

(1) 鍵管理帳の事前配送

予め通信相手との共通鍵を鍵管理帳として設定し、各通信相手へ安全な経路を使用して配送する。また平文を暗号化するにはセッション鍵（作業鍵）を共通鍵暗号アルゴリズムを使用し暗号化して配送する。図 4.2-2 に鍵管理帳の図を示す。

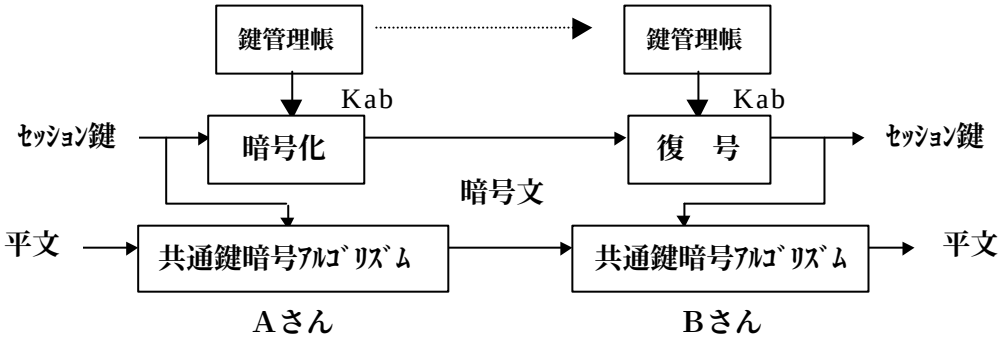


図 4.2-2　鍵管理帳

(2) Kerberos

共通鍵暗号ではいかに送信者と受信者との間で秘密の鍵を配送するかが大きな問題となる。この鍵配送の問題を見事に解決したのが後述の公開鍵暗号であるが、他にもいろいろな解決策が提案されている。その一つが、鍵配送センター（サーバ）を用いた Kerberos である。

Kerberos においては、予め鍵配送センターと各々クライアントおよびサーバとの間で共通鍵暗号の鍵共有を行う。クライアントとサーバとの間の鍵（セション鍵）共有にあたっては、先ず、クライアントが鍵配送センターに相手サーバとの間のセション鍵を要求し、鍵配送センターは要求元クライアントに対して、セション鍵を要求元クライアントおよび相手サーバの各々の鍵で暗号化して返す。要求元クライアントはサーバの鍵で暗号化されたセション鍵を相手サーバに送ることによって鍵を共有することが可能となる。このセション鍵は、鍵管理センターから要求元クライアントに対しては「証明書」と呼ばれるデータ中に含まれており、要求元クライアントからサーバに対しては「チケット」とよばれるデータ中に含まれている。

図 4.2-3 に Kerberos における鍵配送の概念図を示す。

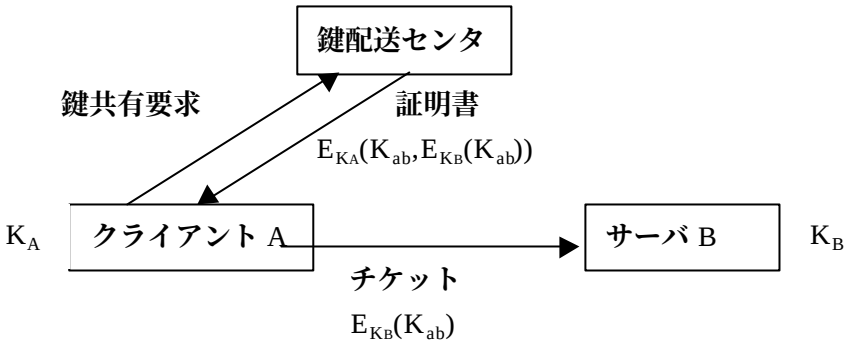


図 4.2-3 にKerberos における鍵配送

4.2.3 公開鍵暗号による暗号化鍵共有

公開鍵暗号系暗号アルゴリズムを使用して、暗号化／復号のときに使用されるセッション鍵（作業鍵）を暗号化して配送する。具体的には、RSA などの公開鍵暗号アルゴリズムを使用して、セッション鍵を相手の公開鍵で暗号化して相手に送り、受信者は自分の秘密鍵で暗号化されたセッション鍵を復号してもとのセッション鍵として鍵共有を行う。

図 4.2-4 に公開鍵暗号系暗号アルゴリズムによる暗号化鍵共有の図を示す。

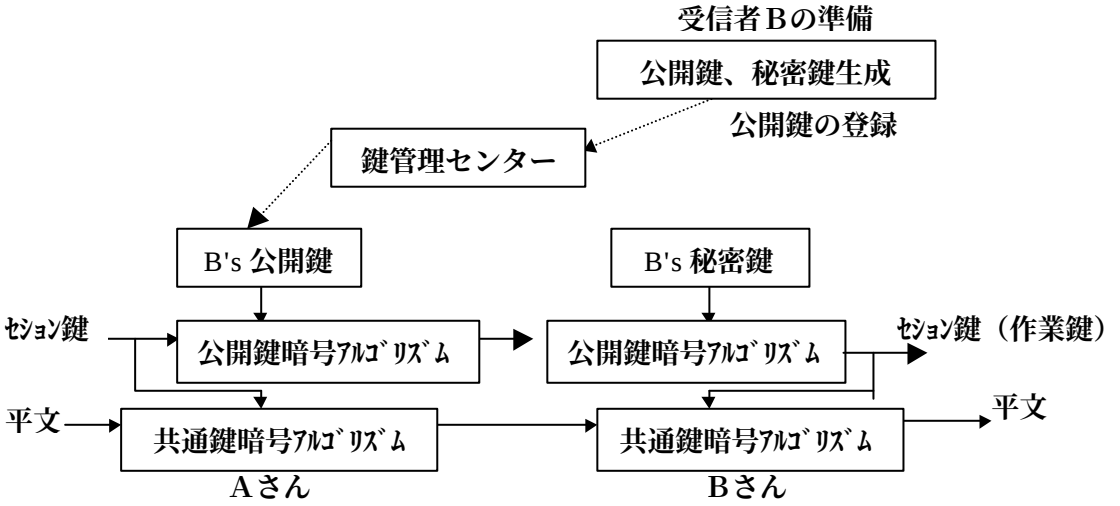


図 4.2-4　公開鍵暗号系暗号アルゴリズムによる暗号化鍵共有

4.2.4 共通鍵生成による暗号鍵共有

この方式は、通信相手相互が、公開鍵等相手の公開情報と自分の秘密鍵情報をもとに共通鍵を生成する。

(1) DH（Diffie-Hellman）による共通鍵生成

1976 年に Diffie と Hellman によって提案された方式で、自分の秘密情報から計算した公開情報を相手に送り、各々相手の公開情報と自分の秘密情報とから共有鍵を生成する。

図 4.2-5 に DH による共通鍵の生成概念を示す

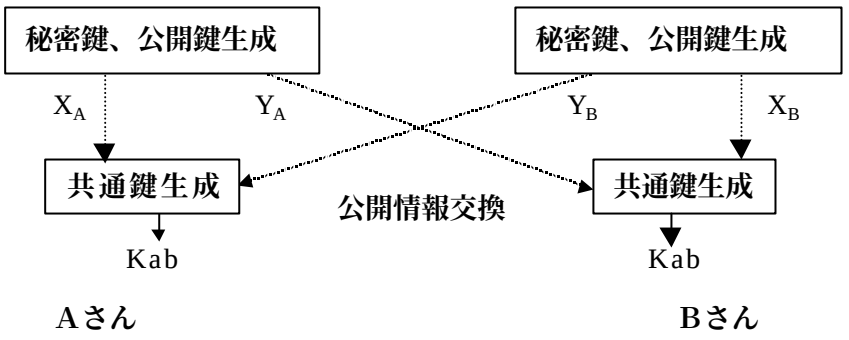


図4.2-5　DHによる共通鍵の生成

(1) KSP による鍵共有

センターからユーザ毎に固有の秘密アルゴリズムを事前に配付しておき、共通鍵生成時に、相手の識別情報（メールアドレスなど）と自分の秘密アルゴリズムとから共通鍵を生成する。

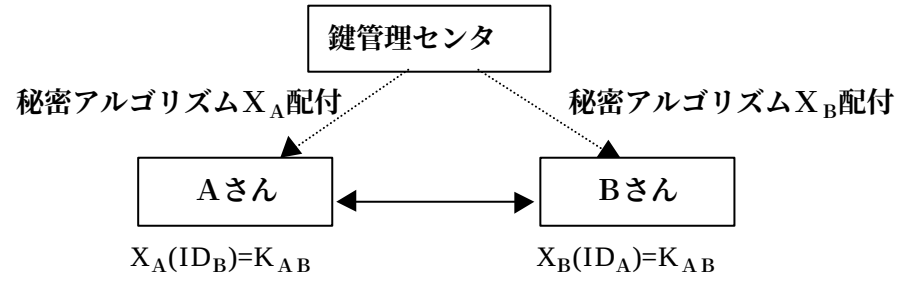


図4.2-6　KSP による鍵共有

4.3 鍵の保管

鍵の保管には、システム設計で一般的に要求される性質（信頼性、経済性、高速性 etc)の他に**耐タンパー性**（それが持つ情報を不正に読んだり、改竄したりするのが難しい性質）が重要である。

4.3.1 物理的隔離

鍵の保管場所を物理的に隔離することで耐タンパー性を確保する。バイオメトリックス（指紋、網膜パターン、虹彩パターン etc）と警備員により厳重な入退室管理が行われてい

る部屋に設置された複数の管理者と鍵でロックされた金庫の中に保管することから、一般事務室の鍵付ロッカーへの保管まで色々なレベルがある。

4.3.2 分散保管

鍵の単純な分割、乱数との排他的論理和による分割、暗号化して鍵と別媒体に分離、特殊な関数による分割などで分割した鍵を複数の管理者または記録媒体に分散して保管することにより耐タンパー性を高める。運用面を含めた耐タンパー性の向上が期待できる。

4.3.3 特殊媒体への保管

(1) 磁気カード

深層記録、特殊な形状のヘッドによる記録、特殊な信号規格での記録、信号の揺らぎの利用、高磁力記録などがある。

(2) IC カード

単なるメモリカードと CPU を内蔵して特殊なプロトコルでやりとりするカードに分類できる。単なるメモリカードでは高い耐タンパー性を期待できない。CPU を内蔵カードの場合は、媒体としての性質でなくそのシステム設計に依存する。

いずれにしても媒体の特殊性が保たれている間だけの耐タンパー性であり、不正な運用・盗難・ロット余り品・破棄品などでリードライト機器や媒体が市場に出回らないよう、不正な管理が無いように配慮する必要がある。

4.3.4 特殊記録フォーマットでの保管

一般的なフロッピィディスクであっても、OS/BIOS で扱えないような特殊な記録方式で記録することで耐タンパー性を高める方法がある。特殊なトラック形式（多数のレコード ID、巨大レコード、極小レコード、単密倍密の混在、異常レコード、etc）、オーバートラック記録、アナログ的な不安定性の導入、媒体に付けた傷など。かつては、高価なビジネスソフトやゲームソフトのコピープロテクトとして盛んに使われた。

しかし、リバースエンジニアリングに耐えられない、ハードウェアが限定される、動作が不安定であるなどの問題点があり EC 関連での採用例は少ない。

4.3.5 暗号技術による保管

パスワードの保管など一致か否かの参照のみで良ければハッシュ化して保管し、暗号化鍵の保管など書いた内容を読む必要が有れば暗号化して記録することで耐タンパー性を確保する。

一例として、ハッシュ化・暗号化の鍵として真性乱数の累積値を使用し、アクセスの都度再ハッシュ化・再暗号化するにより、予測不能な変化の継続で耐タンパー性を高める事ができる。

4.4 鍵管理システム

鍵管理システムは、一般には暗号鍵（共通鍵、秘密鍵）を管理し、攻撃以外の各種の障害（ハードウェアやソフトウェア、事故／天災／人為的ミス／不正行為）などの場合に、

暗号化されているデータを復号可能にするための、インフラストラクチャである。

米国で提唱された鍵回復システム（Key Recovery System）は（政府を含む）第３者機関による鍵の管理を前提とするが、ここでは第３者機関に限らず暗号機能を利用するシステムの構成要素全般に対する鍵の管理と回復のためのシステムをいう。

公開鍵基盤（4.5 参照）インフラストラクチャを利用する場合には、共通鍵の配付や管理に公開鍵暗号を利用することで、このような鍵管理システム自体が不要に思える。しかし、失われた鍵（およびそれにより暗号化された情報）は決して戻らない。暗号化した情報を失わないためには、公開鍵による管理とともに、このような鍵を回復するためのインフラも欠くことが出来ない存在なのである。

そしてなによりも、公開鍵暗号系による鍵管理を行うには信頼できる認証機関が必須であり、認証機関の信頼度（暗号理論面、実装面、処理能力やフォールトトレランス性など）が鍵管理システムの信頼度よりも高くなければならない。

4.4.1 鍵管理システムの必要性

暗号通信のように、仮に暗号鍵が失われたりした場合に、新たな鍵の生成と登録を行うだけで機能が回復でき、過去のデータが残っていなければ、鍵管理システムが無くても運用上の支障にはならない。しかし、暗号化によるデータの保存の場合は、鍵が失われるとその情報が利用不能となる。そのために、重要なデータの保存においては、暗号化に使用した鍵も何らかの手段でバックアップ／保存をしておかなければならない。

たとえば、会社などで個人にそれぞれ鍵を配付（または登録）し利用する場合、その従業員が万一死亡したり、過失やあるいは悪意を持って鍵を故意に紛失した場合、組織上の上位者がそれらを復号できなければ会社としての業務に支障が出るであろうし、場合によってはとんでもない損害を受ける可能性もある。

また、サーバや端末上のプログラムがそれぞれ認証や履歴データの保存などを行う場合に使用する暗号鍵などは、当然その機器内部で暗号化などの手法を用いて保管しなければならない。この時に用いる鍵が、全てのサーバや端末で同じであった場合、万一盗難や解読などが発生すると、全体が危険にさらされるため、通常は異なった鍵を利用しなければならない。しかし、システムの更新や履歴の解析などの際には、これらの鍵を利用する必要があるので、サーバや端末以外の場所に、これらの鍵のバックアップを取っておかなければならない。

なお、マスター鍵となる秘密鍵や共通鍵が、何らかの手段により解読もしくは漏洩してしまった場合、新たな秘密鍵や共通鍵を生成し運用を引き継ぎ、廃棄されたマスター鍵を用いて暗号化されたデータは復号し、新たな鍵で暗号化を行わなければ危険である。この場合、秘密情報でないものに電子署名がなされて、デジタルタイムスタンプが付加されている場合は（たとえば契約書など）、再度署名をし直すとその効力が失われるため、そのままにしておく必要がある。

4.4.2 鍵管理システムの種類

一般的には、鍵を管理するスキームとして集中方式と階層方式、分散方式がある。

(1) 集中方式

ある一つのマスター鍵（もしくは特権）により全ての鍵を管理（または暗号化）す

る方式。管理システムを一個所に閉じ込めることができるので、物理的なセキュリティは保ちやすい反面、地理的に分散した場所に置けないデメリットがある。また、万一マスター鍵や特権レベルが突破されると、全てのシステムに脅威が及ぶ。

(2) 階層方式

鍵管理を階層化し、各階層の鍵はその上位層の鍵により管理（または暗号化）される方式。保管する場所は一個所でも良いし、場合によっては地理的に分散させることも可能である。階層を設定する場合も組織階層を流用することができ、その場合にはたとえば、ある支社で用いられる鍵をその支社長の鍵で管理することが可能である。

地理的に分散させることで、下位レベルの管理鍵や特権レベルが破られても影響はその配下にしか及ばないというメリットがある。ただし、分散したそれぞれの場所で物理的なセキュリティを保つ必要がある。

(3) 分散方式

(1)や(2)の方式で、管理する側の鍵を複数個組み合わせないと、鍵の管理や復号ができない方式。管理すべき鍵を分割して、別々に暗号化する場合と、別々の鍵で何重にも暗号化する方式が考えられる。この方式のメリットは、最上位層で危険分散が可能なため、攻撃や特権レベル所有者の作為による漏洩などが困難な点である。

実際に、鍵回復システムや TTP（信頼される第 3 者機関）などによる鍵の保管では、このような分散方式が利用されている。実際に鍵管理システムとして商品化されている物も、上記各方式およびその組み合わせに対応しているものが多い。

図 4．4-1　鍵管理システムの手法

4.4.3 鍵管理システムのセキュリティ

鍵管理システムを構築・運用する場合は、これらがシステム上の最大の攻撃目標になることを前提にセキュリティを保つための工夫が不可欠である。

認証機関は、情報を公開することが前提なので保持している情報が不正に書き換えられない限りはシステムの安全性は保たれる。一方、鍵管理システムでは、保管している内容が漏洩することがあってはならない。したがって、ネットワーク接続などの危険性を極力抑えた上で、安全な場所に隔離することが望ましい。

最後に、特に鍵管理システムで個々人の使用する鍵を管理することは、プライバシーの問題を常にはらんでいる点に注意が必要である。もしあなたの上司が、業務上の代行処理をするためにあなたの鍵を管理しているとしたら、あなたが個人的に秘密にしている情報までも上司から見られることになる。本当にプライバシーの必要な場合は、業務で利用する場合の鍵とは別に、自分だけが知っている秘密の鍵を用いて暗号化しなければならない。このような注意点を利用者が意識しなければならないとしたら、その暗号システムは十分に活用されずに終わるであろう。

したがって、鍵管理システムの運用、特に鍵の回復時の運用について、あらかじめ十分な制度検討と説明および利用者の理解が不可欠である。できれば、鍵の回復にあたっては①本人の了解（もしくは事前承諾）が必要、②複数の人間による合意が必要、③回復した鍵による復号は（できれば）本人立ち会いで公開の場で行われるべきであること、④明確な理由なく本人が許可した物以外復号しない、⑤プライバシーを侵害した場合のペナルティ、などを運用ルールとして決めておくことが重要である。

4.5 公開鍵基盤

公開鍵基盤(PKI:Public-Key Infrastructure)には色々な定義がありまた時々刻々変化しているが、本書では公開鍵証明書の発行・失効、公開鍵の入手、および公開鍵証明書の有効性確認に関する基盤を提供する技術の総称と位置付ける。

PKI としてよく知られているのが，ISO/IEC/ITU の勧告 X.509[1]である。X.509 では公開鍵と所有者の対応を保証するために CA と呼ばれる第三者機関をおいて公開鍵証明書(証明書)を発行するものとし，CA を共通の信頼点として電子的に認証を行うメカニズムを提供している。そのため，CA 自身が公開鍵暗号の鍵対(公開鍵および秘密鍵)を持ち，証明書発行時にはその秘密鍵で署名してから配布する。

PKIX(Public-Key Infrastructure(X.509))は，このような X.509 の定義に基づいた公開鍵暗号のインフラをインタネットで利用していくために必要なプロトコル群の総称である。PKIX は、インターネット技術における標準化活動の中心的役割を果たす IETF(Internet Engineering Task Force)の PKIX 分科会で検討され、標準化が進められている。

4.5.1 PKIX における PKI モデル

PKIX においてベースとなる PKI モデルを，図 4.5-1 に示す。モデルを構成する各要素について簡単に説明する。

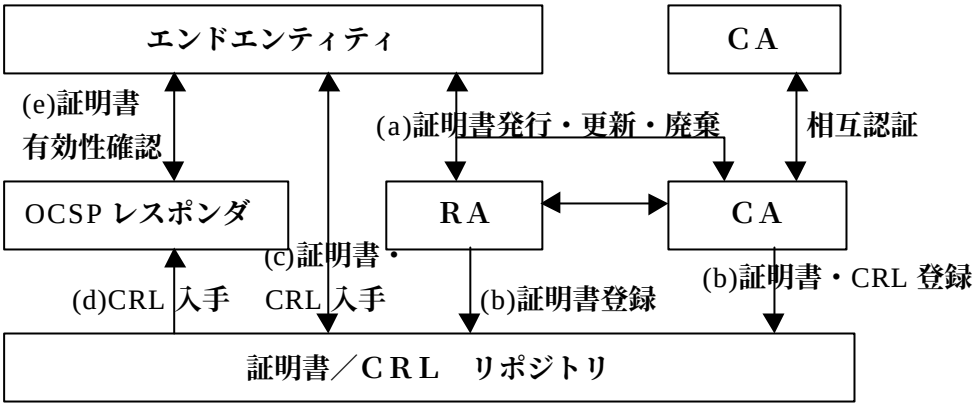


図 4.5-1 PKIX における PKI モデル

(1) エンドエンティティ

証明書の発行対象となる PKI ユーザあるいは PKI ユーザシステム。

(2) CA(Certification Authority)

１つ以上のエンドエンティティにとっての信頼点であり、エンドエンティティの身元確認および証明書や CRL(Certificate Revocation List、証明書廃棄リスト)を発行する機関。

(3) RA(Registration Authority)

証明書発行時の身元確認、審査などの管理作業を行う機関。CA が RA を兼ねることがあるためオプションである。

(4) 証明書/CRL リポジトリ

証明書および CRL を配布するシステム。

(5) OCSP レスポンダ(OCSP responder)

証明書の有効性確認を提供するシステム。

図 4.5-1 のモデルでは，各要素間の基本的な関係が示されている。たとえばエンドエンティティが証明書を発行してもらうためには，まず RA または CA へ要求する(図 4.5-1 の a)。そして，CA が証明書を発行すると RA または CA からリポジトリへ証明書を登録する(図 4.5-1 の b)。エンドエンティティは必要時にリポジトリから他のエンドエンティティの証明書，あるいは CRL を取り出す(図 4.5-1 の c)。

現在，このモデルをオンラインで実現するために必要最低限となる以下の内容が標準化されている。

- 証明書および CRL のフォーマット
  Certificate and CRL Profile[2]。
- CMPCertificate Management Protocols[3]。

(図 4.5-1 の a)に関連するプロトコル。
証明書の発行や更新などの要求のための証明書管理プロトコル。

- OPP(LDAPv2) Operational Protocols - LDAPv2[4]。
  図 4.5-1 の b,c に関連するプロトコル。リポジトリへの証明書の登録やリポジトリからの入手を LDAP(Lightweight Directory Access Protocol)で行うためのプロトコル。

PKIX を利用する際に最も難しいのは、証明書の廃棄管理である。証明書には鍵の寿命や CA の責任範囲などとの関係で有効期限があり、有効期限が切れる前に鍵を紛失したりした場合には所定の手続きにもとづいて CA が証明を取り消す。CA は取り消した証明書のシリアル番号と取り消した日付を管理して定期的に CRL としてまとめて発行し配布する。そして，証明書が現在有効かどうかを検証する時に CRL のデータをつき合わせて電子的に調べることになっている。

PKIX の PKI モデルでは，エンドエンティティが CRL を入手し，証明書の有効性を検証するのが基本である。しかし、１つの証明書の有効性のみを確認したいだけであるのにサイズの大きな CRL を入手したり、毎回 CRL のデータ全体を解析するのは非効率な場合がある。このような問題の解決策の１つとして以下のプロトコルが標準化されている。

- OCSP（Online Certificate Status Protocol）[5]
  エンドエンティティが CRL を持たずに, ある証明書が有効かどうかを CRL を持っている第三者に問い合わせるプロトコル(図 4.5-1 の e)。

4.5.2 証明書および CRL のフォーマット

PKIX では，エンドエンティティが証明書や CRL について正しいかどうかを検証する際に解釈にばらつきがでないよう，X.509 によるフォーマット定義にある程度の制限を加えている。おもな特徴は以下である。

- 証明書では X.509 バージョン 3 対応が必須
- CRL では X.509 バージョン 2 対応が必須
- 証明書のデータに含まれる文字コードとしていくつかの種類を選択可能。

たとえば，UTF8(いわゆる Unicode)を使って漢字を証明書に含めることが可能。

X.509 バージョン２やバージョン３には拡張フィールドが定義されている。証明書については、所有者を他の名前形式で表現する拡張フィールド(subjectAltName など)，鍵の利用目的を示す拡張フィールド(keyUsage)、証明書の利用ポリシを決める拡張フィールド(CertificatePolicies)などがある。

また、CRL では、廃棄理由(reasonCode)などを付加できる。PKIX では，X.509 で定義された拡張フィールドの他，PKIX 独自の拡張フィールドを定義している。そして，各々の拡張フィールドについて，解釈必須の印をつけて使うかどうかを決めている。たとえば，CA の証明書では, CA であることを示す拡張フィールド(basicConstraints)を利用する際には、必ず解釈必須の印をつけるように決めている。

4.5.2 証明書管理プロトコル

(1) CMP

CMP では，エンドエンティティ，RA，CA を含めた証明書管理に必要な機能として以下を挙げている。

- 鍵作成
- 証明書の発行
- 鍵の回復
- 鍵の更新
- 証明書の廃棄
- 証明書発行、更新、廃棄などのアナウンス

これらの各機能を提供するために PKI メッセージを利用した要求および応答フォーマットを定義している。

PKI メッセージは基本的には PKI ヘッダと PKI ボディからなる。PKI ヘッダには送信者や受信者の情報を指定し、PKI ボディには初期化要求や証明書発行要求といった要求の種類ごとに定義されたデータを含める形式となっている。

PKI ボディに使用されるデータフォーマットの多くは、CRMF(Certificate Request Message Format[6])として，CMP とは別のドキュメントで定義されている。証明書の発行要求時の PKI ボディとしては，CRMF で定義されたフォーマットの他、従来から利用されている RSA 社の標準 PKCS10[7]も利用可能である。CRMF では、要求者が公開鍵に対応する秘密鍵を本当に所有していることを確認(Proof of Posession of Private Key)できるようにするためのフォーマットも定義されている。

CMP は特定のトランスポートプロトコルを仮定していないため，HTTP や 電子メールを利用して PKI メッセージをやりとりすることが可能である。たとえば、CMP と S/MIME の摺合わせ案として、PKI メッセージを CMS(Cryptographic Message Syntax)でカプセル化するプロトコル CMC(Certificate Management Messages over CMS)が提案されている。

(2) OPP(LDAPv2)

LDAP 自体は、X.500 やそれ以外のディレクトリシステムにアクセスするための既存のプロトコルである[8][9]。OPP(LDAPv2)では、リポジトリへの証明書や CRL の登録、およびそれらの入手を LDAP バージョン２で行うためにサポートすべきオペレーションを LDAP のサブセットとして定義している。

また、OPP(LDAPv2)とは別のドキュメントとなっているが、PKIX におけるエンドエンティティや CA を LDAPv2 の環境で扱うためのスキーマとして、オブジェクトクラスや属性が決められている[10]。具体的には，エンドエンティティの証明書や CA の証明書を格納するためのオブジェクトクラスや属性、CRL を格納するための属性等が定義されている。

PKIX では、リポジトリにアクセスするプロトコルを LDAP に限定するわけではなく、HTTP や FTP をベースにした OPP(FTP and HTTP)についても標準化した[11]。

(3) OCSP

OCSP は、ちょうどクレジットカードが使えるかを店舗からセンターに問い合わせるのに相当するプロトコルであり、エンドエンティティが CRL を持たずに、ある証明書について現在有効かどうかを知りたい場合に利用される。

OCSP の要求および応答データフォーマットは、CMP のデータフォーマットとは独立に定義されている。要求データには、有効性を確認したい証明書の識別子を指定する。サービス妨害の防止などの目的でオプショナルに要求者の署名をつけることも可能である。応答データには、有効性確認結果、確認した日付、そして無効な場合は廃棄理由などの付加的な情報が含まれ、転送路上の改竄を防止するために署名をつることが必須となっている。そのため、OCSP サービスを行う第三者機関は、応答データに署名するための（公開鍵暗号の）鍵を持つ。

OCSP サービスを行う第３者機関としては、以下の３通りがある。

- CA
- 要求者が（公開鍵を）信用している機関
- CA に OCSP サーバとして認定された機関

これらの機関は、CRL(もしくは CRL よりも頻繁に更新される証明書の状態情報)を管理し、OCSP 応答情報を作成する。OCSP では、特定のトランスポートプロトコルを仮定していないが、HTTP の利用を例として挙げている。

5 暗号の評価と安全性

5.1 暗号アルゴリズムの公開について

セキュリティシステムに暗号アルゴリズムを用いるとき、まず最初に、公開されたアルゴリズムを用いるか、非公開アルゴリズムを用いるかを決めなければならない。このためには表 5-1 のような損得に留意する必要がある。

表5-1 アルゴリズムの公開／非公開に基づくメリット・デメリット

暗号アルゴリズム	メリット	デメリット
公開	評価が明確 アルゴリズムを秘匿しなくてよい	寿命が伸びない
非公開	寿命が長い	評価があいまい アルゴリズムを秘匿しなければならない

以下、この詳細を論じる。

5.1.1 公開／非公開運用形態

暗号アルゴリズムを公開／非公開という運用形態で眺めると、一般に以下のように運用されている。

1. アルゴリズム公開
   DES や RC5 など、誰でも入手できる技術資料としてアルゴリズムが公開されている。DES の後継アルゴリズムである AES もこの範疇に属す。
2. 有償でライセンス取得時に公開
   特許や使用ライセンスの関係上、一般に公開はしないが、システム開発者に対して有償で公開(多くはツールを含めて)される場合がある。
   利用者もライセンス契約により知ることができる。
3. 非公開
   アルゴリズム開発者以外は知ることができない。

5.1.2 各運用形態のメリット・デメリット

現代暗号は、鍵の強さによってのみ暗号の解読を防ぐことを主眼として開発されているので、非公開であるアルゴリズムが暴露されたからといって、即座にそのアルゴリズムを使用する暗号システムの安全性が脅かされるわけではない。よって、アルゴリズムを非公開とする根拠は希薄なようにみえるが、アルゴリズムを非公開運用とする多くの暗号システムが存在するのも事実である。ここでは、運用形態を公開／非公開の２つに大別し、それぞれのメリット・デメリットを述べる。

(1) 公開運用のメリット

1. 総当たり法以外の解読手段での耐攻撃性を検証することが容易である。
2. アルゴリズム自体に問題点が発見されたとき、広く公知され、対策の検討も多く

の協力が得られ、迅速対応が可能である。

このような点で、システム開発者が暗号アルゴリズムの評価に有限な資源しか割け得ない通常の商用システム構築の場合は公開暗号アルゴリズムの使用が一般的となっている。

(2) 公開運用のデメリット

1. 不特定多数による暗号解読技術の向上により、暗号アルゴリズムの寿命が短くなる。

(3) 非公開運用のメリット

1. 公開運用のデメリットの裏返しで、暗号解読技術の向上を防ぐ事ができる。統計的には、公開されて１万人により３年かけて解読法を研究する結果と、非公開で特定の 300 人が 100 年かけて得られる結果は同等である。従って、構築したシステムの寿命を延ばし、開発投資を押さえ、無用なシステム更新を利用者に強いる必要がなくなる。

このような点から、英国などでは政府採用の暗号製品についてはアルゴリズムが非公開であることが前提になっている。また、Skipjack が非公開にされた理由も同様な考えが入っているのであろう。

(4) 非公開運用のデメリット

1. 非公開運用を謳っていても、実際には、関連機器の開発者からの漏洩、リバースエンジニアリング等により実質上公開されてしまう。
2. 多くの人の手による検証がんされていない。たとえ、問題点が指摘されても開発者が肯定も否定もできないため対策が施せない可能性がある。

5.1.3 暗号アルゴリズムの公開と暗号評価

前節の公開／非公開運用のメリット・デメリットで明らかにされたように、ある暗号アルゴリズムが誰もが認めるような暗号評価を得るためには、多くの人手による暗号解読にさらされることが必要になる。そして、これらの暗号解読を生き延びた暗号アルゴリズムが強いアルゴリズムとしての評価を得ている。

一方、アルゴリズムが公開されることにより、暗号評価技術の進展をもたらしたことも事実である。今日の評価技術の多くが、公開暗号アルゴリズムである DES およびその他のアルゴリズムに対する暗号解読技術の成果を基にしている。

5.2 暗号アルゴリズムの安全性

充分に安全性の評価された暗号アルゴリズムの強度は、暗号鍵の鍵長によって規定されるといってよい。従って、セキュリティシステムにおける暗号アルゴリズムの選択においては鍵長の長さをまず決定しなければならない。

A. K. Lenstra、E. R. Verheul は、最近の DES および RSA-512 クラッキングの結果を加味して、共通鍵、公開鍵暗号を問わず 1982 年から 2050 年までのスパンで安全な鍵長の評価を行った。表 5-2 にその結果をしめす[1]。

表5-2 計算量的に安全な鍵長の下限

西暦	共通鍵暗号鍵長 (bit) ※1	公開鍵暗号(RSA、ElGamal)鍵長 (bit)	部分群サイズ (DSA 等) (bit)※2	楕円暗号鍵長(bit) 解読の進歩※3 無	楕円暗号鍵長(bit) 解読の進歩※3 有	実行不可能計算量 (MIPS 年) の目安	H/W コスト 円／日	PentiumII 450MHz での解読時間(年)
1982	56	417	102	105	85	$5.00*10^{5}$	$3.98*10^{9}$	$1.11*10^{3}$
1983	57	440	103	107	88	$8.51*10^{5}$	$4.27*10^{9}$	$1.89*10^{3}$
1984	56	463	105	108	89	$1.45*10^{6}$	$4.57*10^{9}$	$3.22*10^{3}$
1985	59	488	106	110	93	$2.46*10^{6}$	$4.90*10^{9}$	$5.47*10^{3}$
1986	60	513	107	111	96	$4.19*10^{6}$	$5.25*10^{9}$	$9.31*10^{3}$
1987	60	539	108	113	98	$7.13*10^{6}$	$5.63*10^{9}$	$1.58*10^{4}$
1988	61	566	109	114	101	$1.21*10^{7}$	$6.04*10^{9}$	$2.69*10^{4}$
1989	62	594	111	116	104	$2.06*10^{7}$	$6.47*10^{9}$	$4.58*10^{4}$
1990	63	622	112	117	106	$3.51*10^{7}$	$6.93*10^{9}$	$7.80*10^{4}$
1991	63	652	113	119	109	$5.97*10^{7}$	$7.43*10^{9}$	$1.33*10^{5}$
1992	64	682	114	120	112	$1.02*10^{8}$	$7.96*10^{9}$	$2.26*10^{5}$
1993	65	713	116	121	114	$1.73*10^{8}$	$8.54*10^{9}$	$3.84*10^{5}$
1994	66	744	117	123	117	$2.94*10^{8}$	$9.15*10^{9}$	$6.53*10^{5}$
1995	66	777	118	124	121	$5.00*10^{8}$	$9.81*10^{9}$	$1.11*10^{6}$
1996	67	810	120	126	122	$8.51*10^{8}$	$1.05*10^{10}$	$1.89*10^{6}$
1997	68	844	121	127	125	$1.45*10^{9}$	$1.13*10^{10}$	$3.22*10^{6}$
1998	69	879	122	129	129	$2.46*10^{9}$	$1.21*10^{10}$	$5.48*10^{6}$
1999	70	915	123	130	130	$4.19*10^{9}$	$1.29*10^{10}$	$9.31*10^{6}$
2000	70	952	125	132	132	$7.13*10^{9}$	$1.39*10^{10}$	$1.58*10^{7}$
2001	71	990	126	133	135	$1.21*10^{10}$	$1.49*10^{10}$	$2.70*10^{7}$
2002	72	1028	127	135	139	$2.06*10^{10}$	$1.59*10^{10}$	$4.59*10^{7}$
2003	73	1068	129	136	140	$3.51*10^{10}$	$1.71*10^{10}$	$7.80*10^{7}$
2004	73	1108	130	138	143	$5.98*10^{10}$	$1.83*10^{10}$	$1.33*10^{8}$
2005	74	1149	131	139	147	$1.02*10^{11}$	$1.96*10^{10}$	$2.26*10^{8}$
2006	75	1191	133	141	148	$1.73*10^{11}$	$2.10*10^{10}$	$3.84*10^{8}$
2007	76	1235	134	142	152	$2.94*10^{11}$	$2.25*10^{10}$	$6.54*10^{8}$
2008	76	1279	135	144	155	$5.01*10^{11}$	$2.41*10^{10}$	$1.11*10^{9}$
2009	77	1323	137	145	157	$8.52*10^{11}$	$2.59*10^{10}$	$1.89*10^{9}$
2010	78	1369	138	146	160	$1.45*10^{12}$	$2.77*10^{10}$	$3.22*10^{9}$
2011	79	1416	139	148	163	$2.47*10^{12}$	$2.97*10^{10}$	$5.48*10^{9}$
2012	80	1464	141	149	165	$4.19*10^{12}$	$3.19*10^{10}$	$9.32*10^{9}$
2013	80	1513	142	151	168	$7.14*10^{12}$	$3.41*10^{10}$	$1.59*10^{10}$
2014	82	1562	143	152	172	$1.21*10^{13}$	$3.66*10^{10}$	$2.70*10^{10}$
2015	82	1613	145	154	173	$2.07*10^{13}$	$3.92*10^{10}$	$4.59*10^{10}$
2016	83	1664	146	155	177	$3.51*10^{13}$	$4.20*10^{10}$	$7.81*10^{10}$
2017	83	1717	147	157	180	$5.98*10^{13}$	$4.51*10^{10}$	$1.33*10^{11}$
2018	84	1771	149	158	181	$1.02*10^{14}$	$4.83*10^{10}$	$2.26*10^{11}$
2019	85	1825	150	160	185	$1.73*10^{14}$	$5.18*10^{10}$	$3.85*10^{11}$
2020	86	1881	151	161	188	$2.94*10^{14}$	$5.55*10^{10}$	$6.54*10^{11}$
2021	86	1937	153	163	190	$5.01*10^{14}$	$5.94*10^{10}$	$1.11*10^{12}$
2022	87	1995	154	164	193	$8.52*10^{14}$	$6.37*10^{10}$	$1.89*10^{12}$
2023	88	2054	156	166	197	$1.45*10^{15}$	$6.83*10^{10}$	$3.22*10^{12}$
2024	89	2113	157	167	198	$2.47*10^{15}$	$7.32*10^{10}$	$5.48*10^{12}$
2025	89	2174	158	169	202	$4.20*10^{15}$	$7.84*10^{10}$	$9.33*10^{12}$
2026	90	2236	160	170	205	$7.14*10^{15}$	$8.41*10^{10}$	$1.59*10^{13}$
2027	91	2299	161	272	207	$1.21*10^{16}$	$9.01*10^{10}$	$2.70*10^{13}$
2028	92	2362	162	173	210	$2.07*10^{16}$	$9.66*10^{10}$	$4.59*10^{13}$
2029	93	2427	164	175	213	$3.52*10^{16}$	$1.04*10^{11}$	$7.81*10^{13}$
2030	93	2493	165	176	215	$5.98*10^{16}$	$1.11*10^{11}$	$1.33*10^{14}$
2031	94	2560	167	178	218	$1.02*10^{17}$	$1.19*10^{11}$	$2.26*10^{14}$
2032	95	2629	168	179	222	$1.73*10^{17}$	$1.27*10^{11}$	$3.85*10^{14}$
2033	96	2698	169	181	223	$2.95*10^{17}$	$1.37*10^{11}$	$6.55*10^{14}$

2034	96	2768	171	182	227	$5.01*10^{17}$	$1.46*10^{11}$	$1.11*10^{15}$
2035	97	2840	172	184	230	$8.53*10^{17}$	$1.57*10^{11}$	$1.90*10^{15}$
2036	98	2912	173	185	232	$1.45*10^{18}$	$1.68*10^{11}$	$3.22*10^{15}$
2037	98	2986	175	186	235	$2.47*10^{18}$	$1.80*10^{11}$	$5.49*10^{15}$
2038	99	3061	176	188	239	$4.20*10^{18}$	$1.93*10^{11}$	$9.33*10^{15}$
2039	100	3137	178	189	240	$7.14*10^{18}$	$2.07*10^{11}$	$1.59*10^{16}$
2040	101	3214	179	191	244	$1.22*10^{19}$	$2.22*10^{11}$	$2.70*10^{16}$
2041	102	3292	180	192	247	$2.07*10^{19}$	$2.38*10^{11}$	$4.60*10^{16}$
2042	103	3371	182	194	248	$3.52*10^{19}$	$2.55*10^{11}$	$7.82*10^{16}$
2043	103	3451	183	195	252	$5.99*10^{19}$	$2.73*10^{11}$	$1.33*10^{17}$
2044	104	3533	185	197	255	$1.02*10^{20}$	$2.93*10^{11}$	$2.26*10^{17}$
2045	105	3616	186	198	257	$1.73*10^{20}$	$3.14*10^{11}$	$3.85*10^{17}$
2046	106	3700	187	200	260	$2.95*10^{20}$	$3.36*10^{11}$	$6.55*10^{17}$
2047	106	3785	189	201	264	$5.02*10^{20}$	$3.60*10^{11}$	$1.11*10^{18}$
2048	107	3871	190	203	265	$8.53*10^{20}$	$3.86*10^{11}$	$1.90*10^{18}$
2049	108	3959	192	204	269	$1.45*10^{21}$	$4.14*10^{11}$	$3.23*10^{18}$
2050	109	4047	193	205	272	$2.47*10^{21}$	$4.44*10^{11}$	$5.49*10^{18}$

A. K. Lenstra & E. R. Verheul "Selecting Cryptographic Key Sizes"[1]より抜粋

※1 後述するように線形解読法等のショートカット攻撃の bit 換算による効果は 12bit 程度、アルゴリズム処理自体の速度差の bit 換算は 7bit 程度 なので、128bit 長の共通鍵暗号アルゴリズムは今後半世紀間は十分安全であるといえそうである。

※2 ElGamal 署名の署名サイズ、処理量を小さくする手法を採用した DSA 署名では、ElGamal 署名で用いられる素数 p から構成される乗法群の部分群を利用している。この手法は離散対数問題に基づくアルゴリズムに対して一般的な手法である。上記表ではこの部分群のサイズを表わしている

※3 このデータは 1999 年を基に外挿したものなので、1999 年以前の楕円暗号・解読有りのデータは意味が無いことに注意。

A. K. Lenstra、E. R. Verheul の手法は後節で述べる手法と同様、計算量によって安全な鍵長の長さを見積もっている。その基本は４つある

(1) 寿命

どのくらいの期間解読されないかで見積もっている。表 5-2 では 1999 年を基準にしている。

(2) 安全マージン

DES を基準に算出。特に商用ベースでは 1982 年の DES の安全性が基準となる。

(3) 計算機能力

Moore の法則を仮定。

(4) アルゴリズム毎の最適な暗号解読法を想定

後節 5.2.3、5.2.4 を参照のこと

このように、ようやくコンセンサスのとれてきた手法を使って最新(1999 年)の解読データをもとに算出したことが大きな特徴である。

以下に、暗号の安全性について詳述する。

5.2.1　暗号評価と暗号解読

暗号評価の結果として、我々が知りたいことはこれから使おうとしている暗号アルゴリズムが“強い暗号”であるかということである[2,3,4]。

では、この“強い暗号”とは何を意味するかというと、第三者がそのアルゴリズムの詳細を知っていると仮定しても、暗号化に必要な鍵の情報を推定するのに必要な情報量または計算量が十分大きいということに対応する。しかしである、実際にある暗号を“解読”すればこの情報量や計算量は明らかにできるが、この時点でこの暗号は“弱い暗号”になってしまう。一方、我々が必要とする“強い暗号”は解読できない暗号である。そこで、実際に解読を試みなくてもどれくらいの情報量や計算量が必要となるかを評価できる指標、強度評価指標を得ることが重要となる。

このような意味で暗号評価と暗号解読は表裏の関係にあることになる。従って、以下の節で、暗号解読について述べていく。

5.2.2　暗号解読

暗号解読もしくは暗号攻撃は、解読者が暗号アルゴリズムを知っているとの条件で鍵を推定する試みである。通例、以下のような 2 つの前提条件がおかれる。

- 解読者は暗号文を自由に入手できる
- 暗号文は一定の鍵で暗号化されている

さらに、暗号解読者は平文に関しても何らかの情報を持っているとして、その多寡により 4 つの攻撃に分類できる。それを順に述べる。

1. 暗号文単独攻撃
   解読者が平文に関する何らかの統計的情報を持っているが具体的には平文は知らない、という条件での解読。例えば、日本語や英語などの自然言語の統計的偏りの大きさ、定型文からなるヘッダー部は有力な攻撃の手がかりになる。
2. 既知平文攻撃
   暗号文と対応する平文のペアを自由に入手可能という条件下で、鍵を求める解読。暗号アルゴリズムを用いた認証プロトコルを使う環境下ではそのプロトコルを傍受すればよいので、わりかし容易に調う攻撃。
3. 選択平文攻撃
   解読者が任意に選択した平文とその暗号文のペアを手がかりにできる攻撃。既知平文攻撃に比べて解読者がより知りたい情報を入手できる。上記認証プロトコルにおいて、解読者が確認者に成りすませば容易に実現できる。
4. 選択暗号文攻撃
   解読者が、暗号文の方を任意に選択でき、それに対応する平文ペアとを攻撃の手がかりにできる。

特に、公開鍵暗号において、解読者がターゲットに様々な暗号文を送りつけ、それに対する復号結果を手がかりにする“適応的選択暗号文攻撃”は能動的攻撃として最も強力な攻撃と目されている。

5.2.3 共通鍵暗号の評価

これまでに様々な共通鍵暗号が公開されている。こうした公開された暗号アルゴリズムを攻撃することで暗号解読技術は進歩してきた。なかんずくグローバルスタンダードと言ってもよかった公開アルゴリズム DES を中心にした攻撃は多くの成果を暗号評価技術にもたらした。

(1) 総当たり攻撃

最もシンプルで最も強力な解読法に、可能性のある鍵を一つずつ総てチェックする方法がある。この方法は基本的に暗号文と平文のペアが 1 個あればできる既知平文攻撃である。

但し、暗号文の数が多くその統計的偏りが期待できるときには暗号文単独攻撃も可能である。

まずこの方法の限界について述べる。

① 情報量的(無条件)安全性に基づく暗号

ある暗号アルゴリズムの安全性を計る目安として、**無条件安全性**と**計算量的安全性**という二つの概念がある。

1. 無条件安全性

   ある暗号アルゴリズムが、たとえ攻撃者側が無限大の計算機資源を所持していたとしても、決して破ることのできないことが保証されている場合、そのアルゴリズムのことを無条件安全性を持つアルゴリズムと呼ぶ。この場合、情報量的に解読が不可能なので**完全秘匿**(perfect secrecy)ともいわれる。

2. 計算量的安全性

   ある暗号リズムを破ろうと試みる攻撃者が、実際に所持可能な計算機資源をすべて費やしたとしても、暗号が破られるまでに十分な時間がかかることが保証されている場合、そのアルゴリズムのことを計算量的安全性を持つアルゴリズムと呼ぶ

実際に無条件安全性を持つアルゴリズムとして One-time Pad と呼ばれる暗号アルゴリズム(**バーナム暗号**)が知られている。これは、送信者と受信者が通信メッセージと等長の乱数 bit 列を鍵として秘密に共有しておき、その鍵をそれぞれ平文/暗号文と XOR 演算することで暗号化/復号処理を行うというものである。このアルゴリズムでは鍵は真に乱数でなければならず、また、一度利用した鍵は二度と利用してはならない。この後者の性質を反映して、``One-time Pad"と命名されている。

One-time Pad が無条件安全性を持つことは、すでに Shannon によって証明されているが、その理由を数式を使わず直感的に理解するには、以下のように考えてみると良いだろう。もし、攻撃者が無限大の計算機資源を持っていたとすれば、ある暗号文に対して可能なすべての鍵パターンを利用して復号を試みることは可能である。しかし、そのような操作を行ったとしても、彼/彼女の手元に残るのはオリジナルのメッセージと等長の、可能なすべての bit 列であり、その中に唯一含まれる筈のオリジナル・メッセージを選び出すために必要な手がかりは、暗号解読を試みる前と変わらず、まったく得られていないということになるわけだ。

② 計算量的安全性に基づく暗号

ところで、上述の One-time Pad による暗号は、通信メッセージと同じ長さの鍵をもたなければならない点で効率が悪い。逆に通信メッセージよりも短い鍵で暗号化を施すと、無条件に安全というわけにはいかなく、“十分な計算”をほどこせば解読可能である。

現代暗号としての共通鍵暗号アルゴリズムは効率性も追求しなければならないので、短い鍵で多量の平文を暗号化する。従って、どのくらいの計算量が必要かという目安が暗号強度の指標となる。通例次節にのべるような弱点をもたない理想的なアルゴリズムでは、鍵長が具体的な強度指標を与えてくれる。つまり、鍵長が長いほど強い暗号であることを意味し、長さに対して指数関数的に安全性は高くなる。

では現時点においてどのくらいの鍵長であれば安全であるかを確認しよう。

③ 総当たり法による現状とその未来への外挿

1997 年６月 17 日火曜日に、同年１月 29 日より RSA 社が行っていた暗号解読コンテストにおいて、ついに DES が解読された[5]。解読者はユタ州ソルトレイクシティの Michael K. Sanders で、彼は DESCHALL という解読プロジェクトチームの作成した解読プログラムを会社（iNetZ 社）にある FreeBSD を搭載した Pentium 90MHz（16MbyteRAM）のマシンで実行し、正しい鍵 ”85 58 89 1a b0 c8 51 b6” を見つけ出すことができた。

RSA Security 社は全数探索法による DES の解読所要時間を定量的に検証するため、1997 年から DES 解読コンテストを実施してきた。このコンテストでは DES の CBC モードで暗号化された暗号文と初期ベクタが示され、暗号化に用いた鍵を見つけるのに要した時間を競うコンテストである。回を追うごとに解読時間が短縮され、４回目となる 1999 年１月には米国の非営利団体 EFF(Electronic Frontier Foundation)が約 25 万ドルで開発した DES 解読専用装置(DES Cracker)とインターネット上の 10 万台近くの PC を用いて全数探索法により約 22 時間 15 分で解読したことが報告された。EFF は DES 解読専用装置 ”DES Cracker”の技術情報を書籍[6]として刊行しており、誰でも容易に技術情報を入手可能となっている。

表 5-3 DES 解読コンテスト(RSA Security 社)

	コンテスト開始日	解読方法	平均鍵検証個数/秒	解読時間 (鍵全体に対する検証した鍵個数比率)
第 1 回	1997 年 1 月 28 日	約 1 万台の PC	約 17 億個	約 140 日(約 25%)
第 2 回	1998 年 1 月 13 日	約 4 万台*の PC	約 184 億個	約 40 日(約 88%)
第 3 回	1998 年 7 月 13 日	DES Cracker	約 888 億個	約 56 時間(約 25%)
第 4 回	1999 年 1 月 18 日	DES Cracker + 10 万台の PC	約 2,450 億個	22 時間 15 分

DES の鍵は 56bit 長であり、これは数字で表わすと 72,057,594,037,927,936（７京２千兆）にのぼる。

また、56bit RC5 も 10 月 19 日 13:25（GMT）に、Project Bovine と呼ばれるグループにより解読された[5,7]。このように、現在においてはわざわざ超高速のスーパーコンピュータを用意したり、専用の解読装置を何万個も製造することなく、机の上にあるパソコンのアイドルタイムを活用することでも、十分な脅威となりうることが明らかになった。今日では、政府の機関や大企業など数十万台規模の PC をネットワークで接続し、そのほとんどが夜間や土日などには利用されないといった状況のなかで、組織的に（あるいは内部のハッカーに）より同様の解読作業が行われないとは誰もいえない状況にある。

これらのことを考慮すると、現在利用されている 56bit から 64bit の鍵長の暗号方式では、総当たり法に対して十分な強度であるとはいえなくなっている。1996 年 1 月に暗号の専門家を集めたグループが、同様の研究を行っており、やはり 56bit の鍵長の危険性が指摘され、今後 20 年間の期間内で少なくとも 90bit 以上の鍵を利用することを推奨している[8]。

ここでは、上記報告を元に、以下の条件の下での総当たり法での解読時間の推定を行い、暗号の安全性を評価するための基準値を求める。

＜前提条件＞

a)アルゴリズム間の速度の差は bit 数換算で多く見積もって 7 bit 程度である。

b)DES では線形解読法で既知の平文に対して計算量が 12bit 分の 1 程度に少なくなることが知られている。逆にいえば、この程度の強度の冗長性があれば線形解読法は脅威にならないことを示している。

c)各方式は効率の良い解読方法が極力無いように設計されている必要がある。

d)計算機性能は“同一コストで 18 ヶ月で 2 倍になる”Moore の法則“がこれまでも、今後も成り立つものと仮定できる

表 5-4　総当り法による秘密鍵暗号の解読時間（DES56bit 相当）

攻撃者	個人	企業	国家
予算	100 万円	10 億円	1 兆円
解読装置	1000 台の PC 又は FPGA	FPGA/ASIC	ASIC
１９９５年	1.5 年	6 分	0.4 秒
１９９８年	135 日	90 秒	0.1 秒
２００１年	34 日	23 秒	23 ミリ秒
２００４年	8.4 日	6 秒	6 ミリ秒
２００７年	（☆）2.1 日	1.5 秒	1.5 ミリ秒
２０１０年	12.6 時間	0.4 秒	0.4 ミリ秒
２０１３年	3.1 時間	0.1 秒	0.1 ミリ秒
２０１６年	47 分	25 ミリ秒	25μ秒
２０１９年	12 分	6 ミリ秒	6μ秒
２０２２年	3 分	1.5 ミリ秒	1.5μ秒

基準として DES を選択しているので、DES に比べて速度が５倍、10 倍の性能を有する暗号の場合、上記表の５分の１、10 分の１の時間で解読できる事に注意が必要である。これは、DES が１個の鍵を探索する間に、５個、10 個の鍵を探索できることを意味する。従って、DES に換算すると、３bit ないし 4bit 鍵長が短くなったことに相当する。つまり、アルゴリズムが高速であればあるほど安全な鍵長はその分長く準備しておかなければならない。

しかしながら、アルゴリズムの高速化はかなり難しいので、せいぜい 100 倍程度を想定しておけば十分である。これは鍵長に換算すると７bit になる。（前提条件 a)を参照）。

従って、現在の DES56bit 相当の暗号では今後７年程度で、（☆）誰でも数日で解読できるレベルとなることがわかる。

表5-5　今後20 年間情報を守るのに必要とされる90bit 鍵での予測値

攻撃者	個人	企業	国家
予算	100 万円	10 億円	1 兆円
解読装置	1000 台の PC 又は FPGA	FPGA/ASIC	ASIC
１９９５年	257 億年	20 万年	196 年
１９９８年	64 億年	5 万年	49 年
２００１年	16 億年	1.2 万年	12.2 年
２００４年	4 億年	3,064 年	3 年
２００７年	1 億年	766 年	280 日
２０１０年	2,560 万年	191 年	70 日
２０１３年	640 万年	48 年	17.5 日
２０１６年	160 万年	12 年	4.4 日
２０１９年	40 万年	3 年	1 日
２０２２年	10 万年	273 日	6.5 時間

上記の結果より、仮に DES よりも 100 倍（７bit）以上高速な暗号方式であっても、112bit 以上の鍵を使用すると今後 20 年程度なら安全に使用できる可能性があると考えられる。

以上のように、現在の DES56bit はすでに安全性を保証できなくなりつつあり、後継アルゴリズムの多くは、最低でも 128bit の鍵長を準備している。では、どの位まで将来に亙って鍵長は必要になるであろうか。

④　総当たり攻撃法の上限に関する考察

Moore の法則（計算機の速度は 18 ヶ月ごとに２倍になる）が今後も変わらず適用可能だとすれば、無条件安全性を持たない暗号アルゴリズムは、どのような鍵長を用いようと、いずれは破られるということになる。しかし、この仮定は本当に正しいだろうか。

ここでは、さまざまな物理的限界を考慮に入れることにより、現在の手法で解読可能な鍵長に関する上限について考察してみたい。

A．計算機性能向上に影響する要素

まず、計算機性能を向上させる要素が何なのかということを、最初に明らかにしておこう。これは実は単純な話で、主として影響するのは CPU を駆動するクロック数の上昇と、半導体集積度の向上の二つである。

1990 年代後半の Intel 社プロセッサにおける劇的なクロックアップ率は記憶に新しいが、クロックアップはもっとも簡単に計算速度を上げる方法である。同じ論理回路を利用している場合、単純にクロックが 2 倍にできるならば計算機の速度も２倍になる。

また、クロック数が同じ場合は、１クロックでどれだけの計算が実行できるかということが全体の計算速度に影響を与える。同じクロック数を持つ CPU の場合は、複雑な方が全体としての計算速度が速く、そういった複雑さに影響を与えるのが半導体の集積度の向上である(非常に単純化した議論であるが)。また、集積度の向上は、今後は複数の PE(Processing Element)を単一チップに実装するという方向でも進化する可能性があるだろう。

『現状の技術の延長線上で可能な速度向上はほとんど限界に達しているので、近いうちに Moore の法則は成り立たなくなる筈だ』という大方の予想を裏切り、未だに衰えることを知らぬげに CPU の性能は上がり続けている。このままでは本当に現在相当程度に安全な多くのアルゴリズムが、やがては破られる日が来るのだろうか?

さて、このようなクロック数上昇と集積度向上とに影響を与える可能性があるのが、相対性理論および量子論である。さらに、システムを全体として見た場合にある種の upper bound を与えてくれるのが情報理論である。ここでは、これら 20 世紀に次々と発見された物理現象から、理論的にどの程度までの鍵長ならば安全かということを考えてみる。

B. 相対論的限界

相対性理論は、情報の伝達するスピードの上限が光速(2.99792458×10^8m/s)であることを教えている。たとえば、現在得られるもっとも高速な部類の CPU のクロック数は 1GHz 程度のオーダであるため、この１クロックに相当する時間(1nsec)で光(＝情報)はおよそ 0.3m 程度進めることになる。

クロック数を上げるということは、その１クロックで情報が伝達可能な距離も短くなるということを意味する。そのため、たとえば１クロックで原子一つ分程度しか情報が伝達できないという辺りに、可能なクロック数の物理的な絶対限界というものがあると考えられるだろう。

水素原子のボーア半径は 5.29177249×10^(-10)m であるから、その直径をおよそ 10^(-10) m と考えれば、物理的な絶対限界まで、クロックを上げられるのはあと 10^9 倍程度ということになる。これは、暗号技術における鍵長に換算すれば 30bit 程度である。

C. 量子論的限界

さて、それでは集積度を上げる方向ではどのような限界があるだろうか。

集積度を上げる、ということはすなわち LSI などの線幅を細くする、ということと言える。線幅を細くすると、回路中を動く電子にとっては線幅方向の位置に関する不確定性が制限されることを意味する。そして、有名なハイゼンベルグの不確

定性原理

$$\Delta x \times \Delta P_x \geq \frac{\hbar}{2}$$

という関係式より、これは線幅方向の運動量の不確定性が大きくなるということにつながるわけだ。

現在利用されている最新のテクノロジーでは、線幅が 10^{-7}m オーダであるから、そこに電子を閉じ込めるためには、線幅方向の運動量の不確定性は

$$\Delta Px \geq \frac{1.05457266\times10^{-34}}{2} \div 10^{-7} = 5.2728633\times10^{-28}$$

ということになる。

ここで相対論的運動量

$$P = m \times v = \frac{m_0 \times v}{\sqrt{1-\frac{v^2}{c^2}}}$$

から、速度と運動量との関係は

$$v = \frac{c \times P}{\sqrt{c^2 \times {m_0}^2 + P^2}}$$

ここで $P \equiv a \times c \times m_0$ とおけば、

$$v = \frac{a}{\sqrt{1+a^2}} \times c$$

ということは、a=1000 のとき、電子の速度が光速の 99.99995% となることがわかる。おおよそその辺りが物理的な限界と考えられるだろうか。電子の静止質量が 9.1093897×10^{-31} kg であるから、$c \times m_0 = 2.7309263\times10^{-22}$ であるため、前出の運動量の不確定性 $\Delta P_x = 5.2728633\times10^{-28}$ がその 1,000 倍となってしまうまでには、あと 10^9 倍程度線幅を細くできるという程度だろう。これは、暗号技術における鍵長に換算すれば前項と同じく 30 bit 程度である。

D. 情報理論的限界

さて、ここまでクロックアップおよび集積度の向上からそれぞれ実現できる理論的性能向上の可能性を見て来た。最後に考えなければならないのは、そのような限界的な性能向上を果たした計算機を大量に利用するというケースである。力まかせ方式による暗号破りという作業は非常に並列度の高い仕事であるため、個々の計算機には理論的な性能限界があるとしても、そのような計算機が十分多数用意できるとすれば、暗号破りに関する限界は事実上存在しないことになるからだ。

このような場合に上限を与えようと思うと、たとえば地球上に存在するシリコンの量などから考えることもできるが、より興味深いものとしてはエントロピーから考察するという例がある。これは B. Schneier が名著『Applied Cryptography』

[9]のなかで紹介した考え方だが、それを以下に要約してみたい。

さて、Shannon の情報理論は、情報がエントロピーであることを教えている。

温度 T の元で 1 bit の情報を操作するのには kT(k はボルツマン定数 =1.380658×10^{-23}J/K)のエネルギーが必要となる。これは言い換えれば、他のすべての条件が理想的に整えられたとしても、1 bit のカウンタ操作を行うだけで kT のエネルギー損失があることを意味するわけだ。

当面利用可能な最大のエネルギー源である太陽からの熱放射をすべて利用できたと考えよう(たとえば、太陽を囲むダイソン球を構築)。

太陽の総輻射量は 3.85×10^{26}J/s であるため、一年間に太陽が放出する全エネルギーは 1.21×10^{34}J。

これを背景輻射温度 3.2 K の元ですべてカウンタ操作につぎ込んだとして、

$$\frac{1.21\times10^{34}}{1.38\times3.2\times10^{-23}} \cong 2.71\times10^{56}$$

これはおおよそ 187bit に相当する値だ。

これは、つまり、力まかせ方式で対称暗号アルゴリズムの解読を試みる場合、どれほど理想的に環境を整えたとしても、太陽の放出する一年分に相当する全エネルギーをつぎ込んで、やっと 187bit の鍵空間を尽くすのが精一杯ということを意味している。

E. 物理的限界から考えた上限から導かれること

以上のように、各種の物理的な限界から考える限り、対称アルゴリズムにおける鍵長は 百数十 bit 程度あれば十分であると言える。AES は利用可能な鍵長として 256bit まで要求しているが、これはあきらかに過剰仕様であることが明らかであろう。

ただし、以上は暗号解読に関するトレンドが現状のままであることを仮定した場合である。たとえば、前にも述べた通り、計算量に関する理論はまだ十分発達していないため、今後の研究如何では、$P=NP$ であることが証明されてしまうとか、非決定的チューリングマシンが実現するとかいった、現状を根本的に覆してしまうような展開があることも、確率は非常に低いだろうが、あり得ることである。また、それらよりはもう少し実現性の高そうな可能性として量子計算機などというものがかなり精力的に研究されている。これも現在の公開鍵暗号に対しては壊滅的な打撃を与えるものとなる。

ただし、そういったものが実現してしまった場合、『鍵長が 128bit ではダメで 256bit ならば大丈夫』というようなことではあり得ず、暗号技術体系全体に対する根本的な改革が必要となるため、いずれにしても AES の 256bit には意味がないという見解自体には影響なかろう。

(2) 差分攻撃

前節の総当たり法による解読は、暗号アルゴリズムの仕組みに依存しないオールマイティな方法であるので極めて現実的な方法である。しかし、アルゴリズムによってはその詳細な仕組みを知ることで、より高速な解読法(ショートカット法)が見出される。

当節と次節では、代表的あり、かつ、比較的汎用性をもつショートカット法を紹介する。

差分解読法は、'90 年にイスラエルの Biham と Shamir により発表された解読法[4,10]で、選択平文攻撃に分類される。その原理は“平文の変化に対する暗号文の変化を統計的に捉える”ものである。具体的には、解読者は予め平文 P の変化量ΔP(≠0)と暗号文 C の変化量ΔC を固定し、次にランダムに与えた平文 P に対して「平均差分確率」DP：

$$DP(\Delta P \rightarrow \Delta C)=Prob_P\{F(P+\Delta P)+F(P)=\Delta C\}$$

を求める。ここで、F は暗号化関数、+は bit 毎の排他的論理和をあらわす。つまり、平文を少し変更したときに、暗号文も変わるのであるが、その変わり方に癖があるかどうかをみる。特別なΔP、ΔC に対して、この平均差分確率が大きくなるようなアルゴリズムでは、総当たり法よりも効率のよい解読法が存在する。そのてがかりとしての平文・暗号文ペアの数は DP に反比例する。

従って、差分攻撃に対する暗号強度指標は、DP の大きさにより与えられる。厳密には、最大平均差分確率：

$$DP_{max}=max_{\Delta P\neq 0,\ \Delta C} DP(\Delta P \rightarrow \Delta C)$$

の大きさで与えられる。この値が小さいほど強い暗号となる。

但し、この値を実際のアルゴリズムについて求めるためには、膨大な計算量が必要となるので、代わりに最大差分特性確率なる量が用いられることが多いが、この量も正確な値を求めることは難しい。

(3) 線形攻撃

線形解読法は、1993 年に松井が開発した解読法[4,11]であり、既知平文攻撃の範疇に属する。その原理は“平文と暗号文の bit 相関関係を統計的にとらえる”ものである。

具体的には、平文と暗号文のマスク値、ΓP とΓC(≠0)をそれぞれ固定しておき、ランダムに与えた平文 P に対して、以下のような量で与えられる統計的な偏りを手がかりにするのである：

$$LP(\Gamma P \rightarrow \Gamma C)=|2Prob_P\{P\cdot\Gamma P=C\cdot\Gamma C\}-1|^2$$

この値は平均線形確率と呼ばれる。ここで“・”は bit 毎の論理和をとった値のパリティを示す。この値が大きな値をとるアルゴリズムは癖がある弱いアルゴリズムということになり、総当たり法よりも効率的な解読法が存在する。そのてがかりとしての平文・暗号文ペアの数は LP に反比例する。

従って、線形攻撃に対する暗号強度指標は、LP の大きさにより与えられる。厳密には、最大平均線形確率：

$$LP_{max}=max_{\Gamma C\neq 0,\ \Gamma P} LP(\Gamma P \rightarrow \Gamma C)$$

の大きさで与えられる。この値が小さいほど強い暗号となる。

但し、この値を実際のアルゴリズムについて求めるためには、膨大な計算量が必要となるので、代わりに最大線形特性確率なる量が用いられることが多いが、この量も正確な値を求めることは難しい。

この攻撃の差分攻撃との違いは、選択平文攻撃ではなく、既知平文攻撃である点で

ある。

つまり、より攻撃者にとって条件がゆるくても成り立つ攻撃である。実際に、DESに対する初めての計算機による解読実験は線形攻撃によるものであった。

5.2.4 公開鍵暗号の評価

公開鍵暗号の評価も、その考え方として共通鍵暗号の評価と基本的には同じである[2,12]。但し、共通鍵暗号と異なり「公開鍵」を第３者に公開すること、アルゴリズムの仕組みが数学的な構造に由来することにより、攻撃者にとって多くの手がかりが与えられ、攻撃の種類がより多彩になっている。また、解読の程度に関しても、共通鍵暗号が解読できる・できないの二者択一であったのに対し、その構造上一部解読できるという事態も生じる。従って、まず攻撃の種類と解読のレベル、安全性に関する公開鍵暗号ならではの概念について紹介しよう。

(1) 公開鍵暗号の攻撃の種類と解読のレベル

暗号を解読するための計算量は、一般に、攻撃者が利用できる攻撃の種類や、解読のレベルによって変化する。以下に、公開鍵暗号の攻撃の種類と解読のレベルについてを示す。

表5-6 攻撃の種類

攻撃の種類	内容
1)直接攻撃	公開鍵から直接秘密鍵を求める攻撃
2)選択平文攻撃	攻撃者の選んだ平文に対する暗号文を用いる攻撃
3)選択暗号文攻撃	攻撃者の選んだ暗号文に対する平文を用いる攻撃
適応的選択暗号文攻撃	選択暗号文攻撃において、この攻撃で前に用いた平文から、この攻撃で次に用いる暗号文を選択できる場合

表5-7 解読のレベル

ｱ)全面的解読	ある公開鍵に対する秘密鍵を求めることができる場合
ｲ)一般的解読	任意の暗号文に対する平文を求めることができる場合
ｳ)部分的解読	平文の一部を求めることができる場合

このうち、1)及び 2)は、誰でも行える攻撃であるので、これらに対して十分な安全性を有していなければならない。すなわちこれらの攻撃に対して上記解読ｱ)、ｲ)、ｳ)が成立しないようにしなければならない。3)、4)は一般的には成立し難いため、仮にこれらの攻撃に対して安全でないとしても、実用上問題ないケースがある。このように、公開鍵暗号を使用する場合は、想定する攻撃の種類と解読レベルに応じて適切な公開鍵暗号を利用する必要がある。

(2) 直接攻撃に対する安全性

公開鍵暗号では、直接攻撃（公開鍵から直接秘密鍵を計算する攻撃）の難しさが、

素因数分解問題や離散対数問題といった数学問題を解く難しさに対応している。以下では、公開鍵暗号に利用される代表的な数学問題に対してこれまでに知られている解法についてまとめる。以下では計算量の評価に $Ln[a, b] = \exp((b+o(1))(\log p)^{a}(\log\log p)^{1-a})$を用いる。 o(1)は極限値が０の定数。

① 素因数分解問題の解法

素因数分解問題の主な解法を表に示す。表において 1～4 は解読計算量が素因数の性質に依存する解法であり、5～6 は解読計算量が合成数 n のサイズのみによって決まる解法である。n を 2 素数 p, q の積とするとき、(p±1), (q±1)がそれぞれ大きな素因数をもちかつ|p-q|が大きい場合、現時点で最強の解法は数体ふるい法でありその場合の計算量は以下の通りである。

$Ln[1/3,1.922] = \exp((1.922+o(1))(\log p)^{1/3}(\log\log p)^{2/3})$

表 5-8 素因数分解問題の主な解法

解法	内容	計算量
1)試行割算法	合成数ｎを小さな素数から順に割っていく方法	$\sqrt{n}$
2)p-1 法、p+1 法	合成数 n の素因数 p に対して、それぞれ(p-1), (p+1)が小さい素因数の積に分解されるとき有効な方法	p-1 が B-smooth の場合 $B\ln n/\ln B$
3)Fermat 法	合成数 n の素因数 p, q に対して|p-q|が小さい場合に有効な方法	
4)楕円曲線法	Zp 上の楕円曲線 E/Zp の位数(楕円曲線の点(元)の個数)が小さな素数の積に分解できるとき有効な方法	Ln[1/2,1.02]
5)２次ふるい法	２次合同式 $k^2=l^2(\mathrm{mod}n)$を満たす k と l を求め、GCD(k±l,n)より n の素因数を求める方法	Ln[1/2,1.02]
6)数体ふるい法	２次ふるい法を一般化した方法	Ln[1/3,1.922]

② 有限体上の離散対数問題(DLP)に対する解法

有限体上の離散対数問題に対する主な解法を表に示す。

(p-1) が大きな素因数をもつ場合、離散対数問題に対する現時点で最強の解法は数体ふるい法でありその場合の計算量は以下の通りである。

$Ln[1/3,1.922] = \exp((1.922+o(1))(\log p)^{1/3}(\log\log p)^{2/3})$

表 5-9 有限体上の離散対数問題の主な解法

解法	内容	計算量
1)Pohlig-Hellman 法	$y=g^x(\mathrm{mod}p)$において g の位数（ｇが原始根の場合(p-1)）が小さな素数の積に分解されるときに有効な方法	$\sqrt{p}$
2)Index Calculus 法(指数計算法)	ある素数の集合に対する対数をまず求め、それを利用して所望の離散対数を求める方法	Ln[1/2,1.4]
3)数体ふるい法	Index Calculus(指数計算)法の一種で、素因数分解問題に対する最強の解読方法である数体ふるい法を離散対数問題に応用した方法	Ln[1/3,1.922]

③ 楕円曲線上の離散対数問題(EDLP)に対する解法

主な楕円曲線上の離散対数問題に対する解法を表に示す。

MOV-Reduction 法、及び、Smart 及び佐藤－荒木の攻撃法の適用を受けない楕円曲線を利用する場合、楕円曲線上の離散対数問題に対する現時点での最強の解読法は、 Pohlig-Hellman 法であり、その場合の計算量は$\sqrt{p}$ となる。このことから、楕円曲線に基づく暗号・署名方式では、従来の素因数分解問題や有限体上の離散対数問題に基づく暗号・署名方式に比べて鍵サイズを小さくでき、処理を高速に行えると見られている。

表5-10　楕円曲線上の離散対数問題の主な解法

解法	内容	計算量
1)Pohlig-Hellman 法	楕円曲線上のベース点 G の位数が小さな素数の積に分解されるときに有効な方法。	$\sqrt{p}$
2)MOV-reduction 法	Trace=0 の楕円曲線(supersingular)に対して有効な方法。楕円曲線 E/Fp 上の離散対数問題を、有限体上の離散対数問題に帰着させる解法であり、楕円曲線上の離散対数問題が、Fp の小さな拡大体上の離散対数問題に帰着できる。	離散対数問題に対する解法が適用できる
3)Smart 及び佐藤-荒木の攻撃	Trace=1 の楕円曲線(anomalous)に対して有効な方法。素数 p を法とする有限体 Fp 上で構成される楕円曲線 E/Fp においてその楕円曲線の点の個数がちょうど p である場合に有効である。	o(logp)

(3) 主な公開鍵暗号に対する解法

ここでは主な公開鍵暗号について、上記以外の攻撃方法について説明する。

① RSA 暗号の安全性

A. 直接攻撃

RSA 暗号において公開鍵ｎから秘密鍵ｄを求める攻撃法として、ｎを素因数分解してｐ、ｑを求めた後に秘密鍵ｄを求める方法より効率的な方法は今のところ知られていない。したがって、RSA 暗号の全面攻撃に対する安全性は、素因数分解問題に帰着されると考えてよい。

B. 既知平文攻撃

平文 M とそれに対応する暗号文 C を入手し、$M=C^d(\mathrm{mod}n)$から秘密鍵ｄを求める攻撃。合成数 n を法とする離散対数問題に帰着される（この問題は素因数分解問題より困難と考えられている）

C. 短周期攻撃

暗号文 C に対して、f（C）＝$C^e(\mathrm{mod}n)$とするとき、f^t（C）＝C が成り立つとき $M= f^{t-1}$（C）が平文となる。この攻撃に対しては、$(p\pm1), (q\pm1)$の最大素因数をそれぞれ、p_+, p_-, q_+, q_-とするとき$(p_+\pm1)$, $(p_-\pm1)$, $(q_+\pm1)$, $(q_-\pm1)$がそれぞれ大きな素因数を持つとき安全である。

D. 共通 n 攻撃

各利用者が共通のｎを利用するとき、一般的解読（秘密鍵を知ることなく任意の

暗号文に対する平文を解読ずること）が可能となる。この攻撃を避けるため、各利用者はそれぞれ異なる n を用いる必要がある。

E. 小共通 e 攻撃

各利用者が共通の e を利用するとき一般的解読が可能となる。この攻撃を避けるために e は、32 以上とする。

F. 能動的攻撃（実装方式に対する攻撃）

近年になって、適応的選択暗号文攻撃の範疇に属する攻撃者が能動的に解読情報を収集する攻撃法の報告が発表された。ここでは代表的なものを紹介しよう。

a) PKCS#1[13]

1998 年６月、Bleichenbacher は暗号通信プロトコル SSL において RSA 暗号を利用した暗号文が効率的に解読できることを示した。

これはインターネットプロトコル SSL が採用している PKCS#1 というデータ形式に依存した攻撃である。攻撃の種類としては適応的選択暗号文攻撃に分類される。

攻撃者はクライアント-サーバ間の通信を傍受した後、2^{20} 個程度の改竄した暗号文をサーバに送信して、サーバの反応を観測することでオリジナルの暗号文を解読できる。

この攻撃法が発表された後、PKCS#1 は Ver.2.0 に更新された。PKCS#1Ver.2.0 はこの攻撃に対して安全である

b) ISO9796-1，2[14]

ISO9796 とはメッセージ復元型デジタル署名方式の国際標準である。その特徴はメッセージを署名に埋め込むことにより交信データ量が削減できることにある。従って、IC カード上での実装に適している。ISO9796-1，2 は RSA 署名に基づくものであり、旧 ISO9796-2 はフランステレコムの Girault-Misarsky により 1997 に攻撃され、その内容を見直したのち、1999 年９月に標準化された ISO9796-2 は 1999 年４月には、Coron、Naccache、Stern による CNS 攻撃法が発表された。この CNS 攻撃法は変更版 ISO9796-1 にも適用可能であることが発表されている。

Girault-Misarsky 攻撃法は署名検証のために用いられる特殊な関数の性質を利用したものである。この関数に変更を加えたものが ISO9797-2 である。

CNS 攻撃では、任意のメッセージが偽造できるわけではない。しかしながら、1024bit 長の公開鍵の場合、わずか 176bit の因数分解ができればよいので、意味のあるメッセージの署名を偽造することができれば、現実的な脅威となりうる。これもハッシュ関数の特殊な型を手がかりにしている。表 5-11 に攻撃に必要な計算量を示す。

表 5-11 CNS 攻撃法を ISO9796-2 に適用した場合の計算量

ハッシュ長(bit)	必要計算量のオーダー	必要メッセージ数のオーダー
128	2^{54}	2^{36}
160	2^{61}	2^{40}

② ElGamal 暗号に対する攻撃方法

A. 直接攻撃

ElGamal 暗号において、公開鍵から秘密鍵を求める問題はそれぞれ有限体上の離散対数問題に帰着される。

B. 共通乱数ｋ攻撃

同一の乱数ｋで暗号化した場合は、一対の平文と暗号文から同一の乱数で暗号化された平文は解読できる。この攻撃を避けるため乱数 k は毎回変える必要がある。

③ 楕円曲線暗号に対する攻撃方法

A. 直接攻撃

楕円曲線暗号において、公開鍵から秘密鍵を求める問題はそれぞれ楕円曲線上の離散対数問題に帰着される。

B. 共通乱数ｋ攻撃

同一の乱数ｋで暗号化した場合は、一対の平文と暗号文から同一の乱数で暗号化された平文は解読できる。この攻撃を避けるため乱数 k は毎回変える必要がある。

(4) 公開鍵暗号の安全性と鍵サイズとについて

① 鍵サイズと解読計算量について

RSA 暗号、ElGamal 暗号および楕円暗号における鍵サイズと解読計算量の関係を表 5-12 に示す[26]。

ここでは、RSA 暗号、ElGamal 暗号、および楕円暗号に対して、現在知られている最良の解法を適用(5.2.4(2)参照)した場合を仮定している。

表5-12 鍵サイズと解読計算量の関係

ＲＳＡ暗号(bit)	ElGamal 暗号(bit)	楕円暗号(bit)	解読計算量 (MIPS*YEAR)
512	512	100	$3*10^4$
768	768	128	$2*10^8$
1,024	1,024	160	$3*10^{11}$
2,048	2,048	224	10^{21}

ここで MIPS×YEAR は、１MIPS マシン（1 秒に 10^6 回演算を行う能力を有するマシン）を 1 年間動かし続けて得られる計算量であり $3.15*10^{13}$ 回の演算に相当する。

この表は計算量からそれぞれの暗号アルゴリズム鍵長を見積もったものである。従って、計算量のみで安全性等の議論ができる場合には、RSA 暗号の 512bit と

[26]RSA 暗号の 512、768、1024bit における計算量の見積もりは文献[5]に基づく。RSA 暗号の 2024bit、及び、ElGamal 暗号、楕円暗号についても独自に見積もった結果である。

ElGamal 暗号の 512bit と楕円暗号の 100bit は同じようなものとして扱えることを示している。これにより、これ以後、解読時間等を見積もったものの対象を表現する際、それが各アルゴリズムの特性ではなく計算量のみで決定される場合、RSA 暗号を基準にして記述していく。つまり、「RSA512bit 相当」という表現を用いる際は、RSA 暗号 512bit であり、ElGamal 暗号 512bit であり、楕円暗号 100bit が該当する。

② 安全な鍵サイズについて

RSA 暗号、ElGamal 暗号および楕円暗号について、現在知られている最良の解法を適用した場合に、現在から 2022 年までの各時点において、鍵サイズと、暗号解読者の費やす解読コストと、解読時間の関係について検討した結果を表に示す。

なおここでは、次の仮定をおいている。

- 100MIPS のパソコンが約 20 万円で購入できるものとする。
- 1.5 年で CPU パワーが２倍（５年で 10 倍）になるものとする。

表より、RSA512bit 相当の場合、企業レベルの攻撃者を仮定すると、現時点で既に数週間で解読できることがわかる（☆）。また、今後 15 年程度で、誰でも、数週間で解読できるようになることがわかる（☆☆）。

実際、1999 年の８月 22 日ついに RSA512bit が破られたニュースが世界中を走った[5]。これは CWI(オランダ、アムステルダム) の Herman te Riele をリーダとする６カ国にまたがる科学者チームにより、512bit サイズの合成数が素因数分解されたのである。

具体的な方法は、数体ふるい法をにもとずく直接攻撃であり、約 300 台のワークステーションとパソコンが動員されて約７ヶ月で解かれたことになる。計算量的には、約 8000MIPS*YEAR を費やしたことになる。これは先に挙げた表 5-12 に比べて若干小さく、アルゴリズム改良の効果が無視できないことを示す。但し、下表 5-13 の範疇には当てはまっている。

ところで、この計算量は RSA の DES チャレンジで必要とされた計算量の２%にしかすぎなくその意味では時間の問題でしかなかったが、RSA512bit はインターネット上での EC において 95%の割合をしめている(SSL プロトコル等)のでかなり実装の世界では衝撃的なものとなる。今後は、最低でも 768bit、通常で 1024bit の鍵長が要求されていくであろう。

ちなみに、RSA1024bit 相当の場合、企業レベルの攻撃者を仮定しても、今後 20 年間は、解読が困難であることがわかる（☆☆☆）。

また、 RSA2048bit 相当の場合、国家レベルの攻撃者を仮定しても、今後 20 年以上、解読が困難であることがわかる。

表5-13　512bitＲＳＡ相当の場合の解読時間

攻撃者	個人	企業	国家
予算	100万円	10億円	１兆円
１９９７年	60年	22日（☆）	32分
２００２年	６年	2.2日	3.2分
２００７年	210日	5.3時間	19秒
２０１２年	21日（☆☆）	32分	1.9秒
２０１７年	2.1日	3.2分	0.19秒
２０２２年	5.1時間	19秒	0.019秒

表5-14　768bitＲＳＡ相当の場合の解読時間

攻撃者	個人	企業	国家
予算	100万円	10億円	１兆円
１９９７年	$4*10^5$年	$4*10^2$年	150日
２００２年	$4*10^4$年	40年	15日
２００７年	$4*10^3$年	4年	1.5日
２０１２年	$4*10^2$年	150日	3.5時間
２０１７年	40年	15日	21分
２０２２年	4年	1.5日	2.1分

表5-15　1024bitＲＳＡ相当の場合の解読時間

攻撃者	個人	企業	国家
予算	100万円	10億円	１兆円
１９９７年	$6*10^8$年	$6*10^5$年	$6*10^2$年
２００２年	$6*10^7$年	$6*10^4$年	60年
２００７年	$6*10^6$年	$6*10^3$年	6年
２０１２年	$6*10^5$年	$6*10^2$年	220日
２０１７年	$6*10^4$年	60年（☆☆☆）	22日
２０２２年	$6*10^3$年	6年	2.2日

表5-16　2048bitＲＳＡ相当の場合の解読時間

攻撃者	個人	企業	国家
予算	100万円	10億円	１兆円
１９９７年	$2*10^{18}$年	$2*10^{15}$年	$2*10^{12}$年
２００２年	$2*10^{17}$年	$2*10^{14}$年	$2*10^{11}$年
２００７年	$2*10^{16}$年	$2*10^{13}$年	$2*10^{10}$年
２０１２年	$2*10^{15}$年	$2*10^{12}$年	$2*10^9$年
２０１７年	$2*10^{14}$年	$2*10^{11}$年	$2*10^8$年
２０２２年	$2*10^{13}$年	$2*10^{10}$年	$2*10^7$年

5.3 暗号評価とシステムセキュリティ評価

セキュリティシステムを構築する際、如何に安全性の保証された暗号アルゴリズムを用いようともシステム全体の安全性が保証されているとは言い難かった[15]。安全なシステムを構築する技術は、多分にノウハウ的なものからなっており、いわば「安全性品質」というものはシステム毎にまちまちであるといってよかった。これに対して標準的な安全性品質をもたらす期待されるのがコモンクライティリア(Common Criteria, CC)である。従って、今後のセキュリティシステム構築には、CC に基づき安全なシステムを構築していく技術が必須なとなっていくであろう。

※1999 年に CC が ISO 標準化されたことは、設計品質の標準化が ISO9000 シリーズで進められていると同様に、世界的規模で安全性品質の標準化が進められていくことを予見　させる。

では以下に、CC とは何かを詳細にみていこう。

5.3.1 セキュリティ・クライテリア

これまでの節では、暗号アルゴリズム自体の評価を論じてきたが、あくまでも情報システム全体を通して、必要なセキュリティはどのくらいか、そのセキュリティ条件を満たすためには、どの位の安全性を持った暗号アルゴリズムが必要かを見極めなくてはならない。

このためのセキュリティ・クライテリア(安全性基準)の世界的な標準化の動きが、情報ネットワークのグローバル化を追い風にうけて急速に進んでいる。
この中から、代表的なものについてその概要を紹介しておこう。

まず、暗号評価をそのサブセットとして含む「セキュリティ・クライテリア」とは、一般に、情報システムやそれを構成する機器、OS を含むソフトウェアについて、セキュリティ機能全般を、統一した基準に基づいて評価し、その結果でセキュリティレベルを格付けする仕組みである。
このセキュリティクライテリアは、基本的に、

1. 評価基準
2. 認証制度
3. ガイドライン

の３つから構成されることになる。

広義のセキュリティ・クライテリアは、決して新しいものではなく、その規模により、

1.アワード：いわゆるベストプロダクツ賞のようなもの
2. プライベートクライテリア：各企業群、各社が定めるもの
   ICSA 社
   日本ベリサイン：CPS(認証局運用規定)
3. ナショナルクライテリア：国家が定めるもの
   FIPS：米 NIST
   TCSEC：米 NCSC

4.インターナショナルクライテリア：国際間で通用させていこうというもの

ITSEC

Common Criteria(CC)

に分類される。

特に、ここで、暗号製品に関して、我々となじみが深いナショナルクライテリアである NIST の定める FIPS、ことに暗号モジュールを対象にした FIPS140-1 と ISO 標準化も決定し(1999 年 6 月 ISO/IEC15408)、今後一層結び付きが強くなっていくと思われるインターナショナルクライテリアである CC(コモンクライテアリア)について触れよう。

5.3.2 FIPS140-1

FIPS140 は、NIST が管轄する暗号モジュール認証制度である。米軍および米連邦政府のコンピュータ及び電気通信システムにおける unclassified(機密扱いではない)データを対象とする暗号技術応用製品の使用標準を与えている。

この制度は５年に１度見直しが行われ、現在は 1994 年に改定された FIPS140-1 にあたる。1999 年は改定年にあたり FIPS140-2 の登場が待たれている。

歴史的な経緯をふりかえると、もともとは、1982 年に米 NSA による DES の規格・認証プログラム FS1027 として制定されたものが、1988 年に米 NIST に移管され FIPS140 となった。これが前述のように 1994 年に FIPS140-1(暗号モジュールのセキュリティ要件)に改定され、更に、1995 年には他の規格：FIPS46-2、81 (DES、および その操作モード)、FIPS186、180-1 (デジタル署名標準 DSS および ハッシュ標準 SHS)と併せて、米 NIST、加 CSE による暗号モジュール認証(CMV)プログラムがスタートした。そして、現在に至っている。

この暗号モジュール認証制度の仕組みは、まず暗号製品のベンダが民間の会社からなる「評価ラボ」に製品の認証申込を行う。評価ラボでは、暗号製品を評価し、評価レポートをアメリカなら NIST、カナダなら CSE に送る。NIST もしくは、CSE はこの評価レポートに基づき暗号製品の認証をベンダに与える。併せてバリデーションリストを作成し、ユーザ向けに公開する。また、NIST ないし CSE は民間会社に評価ラボの認可を与えることもする。この認可は NVLAP(National Voluntary Accreditation Program) という認可基準に基づき与えられる。

さて、FIPS140-1 の内容であるが、大きく以下の３つから構成される：

1. Rating(格付け)
2. Security Requirements (セキュリティ要件)
3. Testing Requirements (テスト要件)

1. Rating とは、いわゆる格付けであり、４つのレベルからなる。
2. Security Requirements には、以下の 12 の項目がリストアップされている。

①ドキュメント　②インターフェース　③役割とサービス

④状態遷移　⑤耐タンパ　⑥EFP/EFT
⑦ソフトウェア/ファームウェア　⑧OS　⑨鍵管理
⑩暗号アルゴリズム　⑪漏洩電磁波　⑫自己診断

3. Testing Requirements は上記項目それぞれに個別に定義されていて、その結果として各 Rating が決まるようになっている。その概要を示すと、以下の表のようになる

表5-17 FIPS140-1 の具体例

レベル	1	2	3	4
・インターフェース			秘密パラメタ用ポートの物理的分離	
・耐タンパ	通常製品レベル	痕跡が残るシール	検出&秘密消去	同左(異常環境)
・OS	シングルユーザ	C2	B1	B2
・言語			高級言語	述語論理言語
・鍵管理	手動の平文配送可、自動は暗号化		手動・自動とも暗号化配送	

※OS のレベルは、TCSEC が定めるレベルである。

Validation List としては、1998 年４月 15 日の時点で FIPS140-1 認証製品が以下のように、

レベル１：４社８製品

S/W：Entrust 社(Cryptographic Module V1.9，Cryptographic Kernel V2.4)

H/W：Cylink 社(Turbo Crypto Card V09)，Transcrypt International 社(SC20-DES V1.0)，Motorola 社(ASTRO Subscriber，Radio Network Controller，ASTRO-TAC Digital InterfacenUnit，ASTRO XTS 3000 Subscriber)

レベル２：６社７製品

S/W：Netscape 社(Security Module 1)

H/W：IRE 社(SafeNet/LAN VPN Encryptor，SafeNet/Dial Secure Modem)

カード：National Semiconductor 社(Fortezza PCMCIA)，SPYRUS 社(FORTEZZA Cryptocard V0.2)，Mykotronx 社(Palladium Fortezza Crypto Card)，Chrysalis-ITS 社(Lunal PCMCIA Token)

レベル３、レベル４：無し

挙がっている。また、暗号アルゴリズム認証製品としては、

DES：115 製品

DSA・SHA-1：19 製品

が挙がっている。

レベルの概要は以下の通り、

レベル１：

・FIPS 認定アルゴリズムの使用、ここで、FIPS 認定アルゴリズムとは DES，DSA，SHA-1 の３つである。

・物理的セキュリティ対策はない。

レベル２：

レベル１に加えて、以下の３条件が付加される、

・物理的セキュリティ対策：タンパ痕跡コーティング/シルード/ロック

・マルチユーザでの使用は C2 クラスの OS
・役割ベースのオペレータ認証

レベル３：

レベル２に加えて、以下の４条件が付加される。
・タンパリング検出で秘密パラメタの消去
・秘密パラメタポートとデータポートの分離
・マルチユーザでの使用は B1 クラスの OS
・本人認証によるオペレータ認証

レベル４：

レベル３に加えて、以下の２条件が付加される。
・温度、電圧などの異常環境検出で有効化する耐タンパ機能
・マルチユーザでの使用は B2 クラスの OS

※C2 クラスでは、ログイン手続き、セキュリティ関連イベントの監視、リソースの分離を通してユーザのアクセス管理を行っている。B1 クラスでは、C2 クラスに加えて、任意のセキュリティポリシー、データラベリング、強制アクセス制御がサブジェクト、オブジェクト上に付加されている。B2 クラスでは公式のセキュリティポリシーとなり、すべてのサブジェクト、オブジェクトに任意及び強制アクセス制御が課される。

5.3.3 CC(Common Criteria)

まず、これまでのセキュリティ評価・認証の動きを振り返ると、もともと、軍需、政府がセキュリティレベルの高い製品を安価に調達しようとするための制度であったものが、最近になり新しい動きを見せてきた。

１つは、国際的統一の動きである。1999 年６月に Common Criteria が ISO/IEC15408 と標準化された。このセキュリティ要件の各製品カテゴリー毎のサブセットも PP(Protection Profile)として ISO に登録化が進められている。

２つは、欧米６カ国（アメリカ、カナダ、フランス、イギリス、ドイツ、オランダ）によるセキュリティ評価手法の統一基準作りとオランダを除く５カ国間相互認証協定締結である。

結果として、これが CC となり ISO 標準化された。

３つは、この評価・認証が軍需、政府調達を越えて民間システムに適用範囲を拡大しようとしている。

４つは、軍需品から民需品への評価対象の転換に伴い、民需品に適した評価手法の探索と制定。

つまり、セキュリティ評価手法の評価基準としての標準化、評価認定としてのグローバル化、評価対象としての汎用化ニーズの結果として、CC という評価基準とその運用制度が世界的規模で確立されつつある。従って、我が国においても速やかにこの枠組をとりこんでいくことが必要になっている。

ではこのセキュリティ評価基準 CC の概要をみると、FIPS140-1 の構成と同様に３つの

軸から構成されている：

1. 保証レベル(EAL)
2. 機能要件
3. 保証要件

である。

この保証レベル EAL(Evaluation Assurance Level)と米国 TCSEC（オレンジブック）、欧州 ITSEC でそれぞれ定義されているセキュリティ保証レベル（クラス）との対応関係は以下のようになる。

表 5-18 保証レベル(EAL)

レベル	名称	米国 TCSEC	欧州 ITSEC
		D	E0
EAL1	functionally tested		
EAL2	structurally tested	C1	E1
EAL3	methodologically tested and checked	C2	E2
EAL4	methodologically designed, tested and reviewed	B1	E3
EAL5	semiformally designed and tested	B2	E4
EAL6	semiformally verified design and tested	B3	E5
EAL7	formally verified design and tested	A1	E6

なお、米国 TCSEC の保証レベルは CC および ITSEC と枠組みが異なるため単純に比較できないことを付言しておく。言い換えると表 5-18 はあくまでも対応関係にしかすぎない。

保証レベル EAL3 または EAL4 が、民需品が備えるべき保証レベル（米国 TCSEC の C2～B1 相当）と言われている。しかしながら、CC という制度自体が新しいということもあり、この保証レベルをクリアしている製品は殆どないというのが現状である。

なお、軍用あるいはそれに準ずる用途向きは EAL5 以上が要求されている。

機能要件は、製品のセキュリティ機能を規定するもので、以下の 11 クラスに大分類される。

表 5-19 機能要件クラス

FAU	監査
FCO	通信/否認拒否
FCS	暗号
FDP	データ保護
FIA	識別・認証
FMT	セキュリティ管理
FPR	プライバシー
FPT	セキュリティ機能保護
FRU	可用性・リソース管理
FTA	アクセス制御
FTP	高信頼性経路

これは、更にファミリと呼ばれる 66 個の中分類、コンポーネントと呼ばれる 135 個の小分類へと細分化されている。要件細目でみると 250 項目になる。

保証要件は、製品が備えるべきセキュリティ機能要件が正しく実装され、利用できる状況にあるかをどの程度厳格に確認すべきかを規定。以下の 10 クラスに大分類される。

表 5-20 保証要件クラス

APE	PP 評価
ASE	ST 評価
ACM	構成管理
ADO	配布・操作評価
ADV	開発/実装評価
AGD	ガイダンス文書
ALC	ライフサイクルサポート
ATE	テストの評価
AVA	脆弱性評価
AMA	保守

これは、44 ファミリの中分類、96 コンポーネントの小分類に細分され、783 項目の要件細目からなっている。

つまり、保証レベル（EAL）とは、上述した保証要件をどの程度満たしているかにより決定されるのである。

ここで、ある製品について、

- 想定した使われ方と脅威
- セキュリティ目標
- 必要とするセキュリティ機能
- 目標とする保証レベル(EAL)
- 目標とする保証レベル(EAL)に対応する実施すべき評価(保証要件)

をまとめた文書を PP(Protection Profile)あるいは ST(Security Target)と呼ぶ。PP と ST の違いは、PP がある製品カテゴリーについて定義したものに対して、ST は個別具体的製品に対して実際の評価を行うために作成される点である。

従って、この CC を利用した評価・認証シナリオは、まず、CC と、保証レベル(EAL)選択ガイドライン、その他追加要件等をもとに、ユーザ要求に相当する PP(Protection Profile)が製品ジャンルに対応した CC のサブセットとして作成される。次に、この PP と目標から ST(Security Target)と呼ばれる製品に合わせて CC を具体化したセキュリティ仕様書を作成する。この ST をもとに製品開発が行われ、製品(TOE)と開発・製造ドキュメントが評価試験対象物としてベンダから提出される。この提出物について評価試験がおこなわれる。(参照：表 5-21 CC、PP、ST の相違)。この試験に合格すれば、目標とする保証レベル(EAL)の評価が認定される。

表 5-21　CC,PP,ST の相違

	CC	PP	ST
適用	すべての情報システム	製品/システムのカテゴリーに対して	個別の製品/システムに対して
内容	現在考え得るセキュリティ要件のすべて	ユーザの立場で必要とするセキュリティ要件 ＋それを利用する上での前提条件	ベンダが製品に搭載したセキュリティ要件 ＋それを利用する上での前提条件
目的	共通辞書の提供	ユーザニーズの記述 (ST 作成に引用できる)	製品仕様の記述(実際のセキュリティ評価はこれに基づいて実施する)
セキュリティ要件の範囲	フルセット(カタログ)	CC から必要な要件を抜粋したサブセット＋追加要件	PP を参照した 要件あるいは CC から必要な要件を抜粋したサブセット＋個別追加要件
対象との関連	製品/システムに依存しない	製品/システムカテゴリに依存するが実際の製品/システムへの実装には依存しない	実際の製品/システムへの実装に依存する(実装と完全に対応する)
構成	Part1 概要 構成、モデル、 使用法、PP/ST 規定 Part2 セキュリティ機能 ・セキュリティ機能要件 11 クラス 66 ファミリ 135 コンポーネント (トータル 250 項目) (互いに独立、組み合わせ要注意) ・適用ノート Part3 セキュリティ品質 ・セキュリティ保証要件 10 クラス 44 ファミリ 96 コンポーネント (合計 783 項目) (上位は下位を包含) ・評価保証レベル(EAL)に対する保証要件 (セットの定義)	PP 序説 PP の ID、名称 PP の要約説明 TOE(評価対象)の記述 TOE のセキュリティ環境 使用上の前提条件脅威 組織のセキュリティポリシー セキュリティ目的 TOE のセキュリティ目的 使用環境に依存するセキュリティ目的 IT セキュリティ要件 TOE セキュリティ機能要件 TOE セキュリティ保証要件 使用環境に依存するセキュリティ要件 PP 適用の参考情報 論理的根拠(セキュリティ目的、要件)	ST 序説 ST の ID、名称 ST の要約説明 CC 適合性記述 TOE(評価対象)の記述 TOE のセキュリティ環境 使用上の前提条件脅威 組織のセキュリティポリシー セキュリティ目的 TOE のセキュリティ目的 使用環境に依存するセキュリティ目的 IT セキュリティ要件 TOE セキュリティ機能要件 TOE セキュリティ保証要件 使用環境に依存するセキュリティ要件 TOE セキュリティ仕様概要 PP 宣言(引用、追加、修正、詳細化) 論理的根拠(セキュリティ目的、要件、TOE 仕様、PP 宣言)

ここで、一般に PP は製品に特化されないため同種類の製品に対し業界団体などで共通要件を定義している（IC カードなど）。各ベンダはこの PP を参照して ST を作成する。

PP が存在しない場合は PP に記述する内容すべてを ST に記述する必要がある。

一方、TCSEC,ITSEC における評価試験をベースに CC の基準に沿って評価手法を標準化する活動が始まっている。

また、独、英、カナダ、オーストラリア、ニュージーランド、米、および仏は相互認証に合意しており、どの国で評価してもその評価結果が認められる。

6 暗号システムの実装・構築方式

暗号ライブラリやアプリケーションを利用したシステムの構築で、トータルなシステムとしての安全性が必要である、本章ではシステムを構築していく上での留意点などをまとめる。

6.1 暗号システムの構成

6.1.1 実装上の留意点

暗号システムの構築においては、脅威となる対象はなにかを分析し、また守りたいデータの伝送経路をよく把握し、どこでデータが露出するか、また最終的な保管はどのようになるかをよく知る必要がある。以下に実装における留意点を示す。

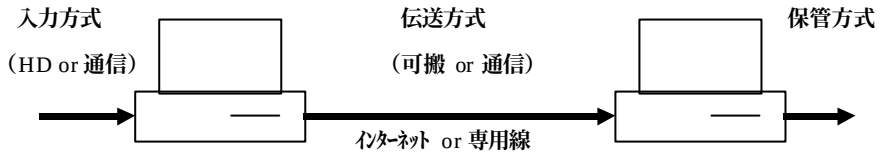


図 6.1-1　データ伝送、保管の例

- パスワードなどの認証情報を入力する場合、その設定は第三者が推測し難いものにする。
- 通信路がインターネットなどの回線では、暗号鍵の長さが長い暗号アルゴリズムを用いる。（但し、輸出規制を考慮する必要がある）
- 出力側では、暗号化されたままの状態で保管し、必要に応じて復号することが望ましい。

また、運用においても、認証情報の設定値や、データ保管形態（媒体、保管場所、扱者特定など）に注意すべきである。

また、暗号化/復号を行う部分や鍵の生成などは暗号ライブラリとして提供されているが、暗号化/復号を行うための鍵を記憶する媒体や方法、鍵の配送方法など暗号ライブラリの入出力部分は新規に開発する場合があり（特に共通鍵暗号+秘密鍵方式）安全性に対し十分考慮する必要がある。

(1)　暗号鍵と暗号文の保存

暗号文と暗号鍵を同一の媒体に保管してしまうことで、第三者が容易に復号可能となる。したがって、暗号鍵と暗号文は離れた場所とするか、暗号鍵を暗号することが望ましい。

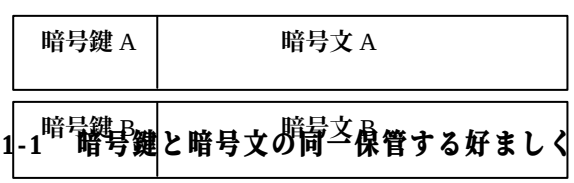


図 6.1-1　暗号鍵と暗号文の同一保管する好ましくない例

また最近の暗号ライブラリでは、ハードウエアインターフェースが提供され、鍵を

PCMCIA カードなどハードウエアに保存するものもある。

(2) 暗号鍵を推測しにくくする

ユーザ ID などを直接使用する場合や、単純なスクランブルを施したものを暗号鍵として使用すると、暗号鍵が推測されやすくなり暗号文を第三者に解読される恐れがある。

また、暗号鍵は別の鍵で暗号化し保存する方法が一般的に用いられている、別の鍵とは公開鍵暗号方式を用いる場合や、パスワードを別の鍵のもととして一方向性変換したものなどを使用する（4.2 鍵の配送を参照）。

(3) 暗号鍵は乱数性のあるもの（または乱数、擬似乱数）を使用する

暗号化を行う場合の暗号鍵は、暗号化毎異なる乱数性を持ったものを使用し、上記(2)で示したように、暗号鍵を他の鍵で暗号化する。

また、暗号ライブラリでは暗号化に擬似乱数を使用しているものが多い。

擬似乱数：算術などの演算により得られた乱数で何らかの規則性があるもの乱数（真性乱数）：不規則、平等な確立、前後が無関係、周期が無いなどの性質を持ち算術乱数では発生できない。

(4) 暗号化にはそれぞれ個別の暗号鍵を使用する。

多くの暗号文を一つの暗号鍵で生成することで、暗号文から鍵が推測され易くなるので注意が必要である（４章参照）。また、暗号ライブラリで鍵シールデイング・ルーチンが提供されている場合もある。

(5) パスワードなどの認証情報

パスワードなどの認証情報は、同一文字の連続（例：AAAA･･･）や連続数字（例：1,2,3,･･）を使用させないように、パスワードの設定方法を考慮し構築する。

上記(1)～(5)に対応した共通鍵を使用したファイル暗号化システムについての鍵管理例を図 6.1-2 に示す。

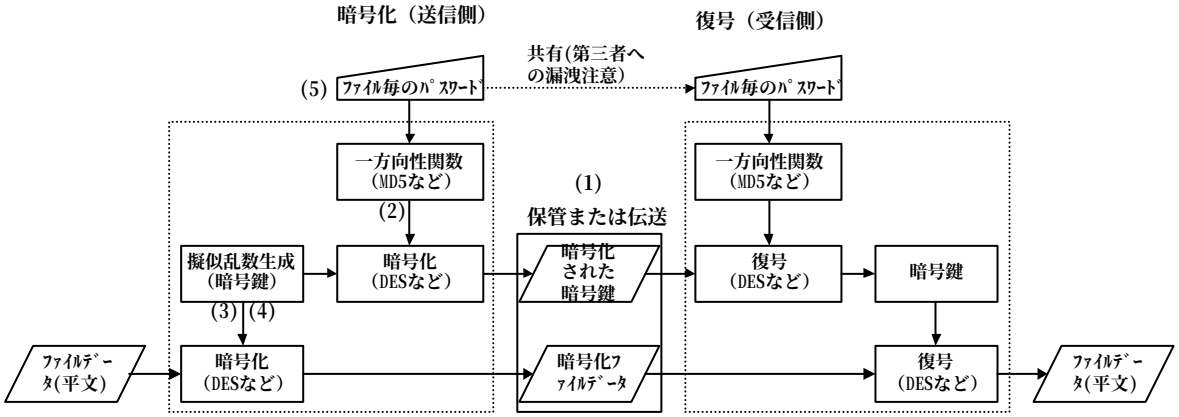


図6.1-2　ファイル暗号化システムでの鍵管理の例

6.1.2 ソフト及びハードウエア構成

暗号ライブラリでは一般的に関数が用意されており、これを使用してプログラムを作成する。例えば、図 6.1-1 に示すように、メインプログラム中で暗号化を行いたい箇所で暗号ライブラリを呼ぶための関数名を書きコンパイルを行う。

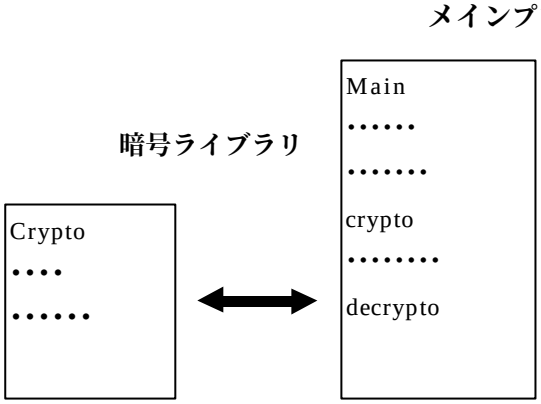


図 6.1-3　暗号ライブラリ参照の例

ASIC などのハードウエアでは各回路毎にまとめてあるブロック構成が一般的であり、他の回路が暗号ブロックとして利用する。

暗号ブロックの実装は、暗号ライブラリに添付されている暗号プログラムを参考にし、回路またはハードウエア記述言語（VHDL や VerilogHDL）に変換し実装する場合や、暗号アルゴリズムから回路を検討する場合がある。

6.1.3 識　別

暗号システムでは、装置間で何らかの暗号データを通信している場合が多く、相手の装置が正しいものか、装置自体や内部が入替えられたものでないか（偽のハードウエアやソフトウエア）を検査するための識別が必要である。例えば、暗号文を解読しなくても通信路を流れている暗号データの関係を知るだけで、偽の装置を作れる可能性もある。

したがって、装置固有の情報を利用して認証情報を生成し相手を識別する。

ハードウエアからは、CPU-ID や BIOS の ver、製造年月日などの固有情報が得られる、またソフトウエアでは、自分自信が改竄されていないかの判別を行う（ハードウエア上の固有情報とソフトウエアのハッシュ値を取り、センターに登録されているものとの比較を行うなどの方法が考えられる）。また、ソフトウアエアを実行するにあたり、上位プログラムや DLL などを使用する場合もあり、その対象プログラムの識別のため固有情報（ver, 日付,容量,その他）も知る必要がある。

6.2 暗号ライブラリ

暗号ライブラリは、暗号処理や乱数生成などの基本機能を持つものや、認証機関構築用のツールキットのように上位アプリケーションを構築しやすいツールもある。

また、これらは PKCS(Public-Key Cryptography Standards：RSA Security 社が提唱する暗号に関する業界標準規格)や PKI(Public Key Infrastructure：4.5 公開鍵基盤参照)に準じているのが一般的である。

6.2.1 暗号ライブラリの構造

(1) 基本ライブラリ

一般的には、暗号エンジン、乱数生成、フォーマット指定、エンコード処理、などを含んだものが多く、C言語や、JAVA 言語などで記述されている。

図 6.2-1 に代表的な基本ライブラリの構造を示す。

表6.2-1　基本ライブラリの構造例

API	暗号選択、パディング選択、その他
暗号エンジン	Rivest,Shamir,Adleman により開発された方式、楕円曲線,DES,DH,RC2,RC5,MD5,SHA-1,etc
鍵生成、乱数生成	鍵生成、擬似乱数生成
フォーマット	ヘッダ(暗号化鍵情報 etc),PKCS#1,5,8,12
メモリ管理	メモリ制御、メモリ使用最適化
高速数値計算処理	数値演算用エンジン
データエンコーディング	Base64,PEM,BER
ハードウエアインターフェース	ハードウエア API、鍵の保存

(2) 認証機関構築ツール

大規模ＣＡ構築ツールやプライベート CA 構築ツールがあり、用途に応じて使用する。企業内での証明書発行などは、プライベート CA 構築ツールを使用し、大規模な証明書の発行は証明書の階層構造がとれる大規模 CA 構築ツールを使用する。

これらのツールの多くはC言語での記述されおり、基本ライブラリを必要とするものと、構築ツールに基本ライブラリの機能が含まれているものとがある。

次に代表的な認証機関構築ツールの主な関数を示す。

6.2.2 主な種類と特徴

(1) 基本ライブラリ

代表的な基本ライブラリを表 6.2-1 に示す。（順不同）

表6.2-1　基本ライブラリ

注：各社製品カタログ及びＷｅｂを参照し記述

名　　称	メーカー	アルゴリズム
BSAFE CryptoC/J	RSA Security 社	RSA,EC,ECDH,ECDSA,DH,DSA,DES,RC2,RC4,RC5,MD,MD2,MD5,SHA-1
PowerMISTY	三菱電機(株)	MISTY,DES,DES-EDE2,DES-EDE3,RC2 相当，楕円 ElGamal,Rivest,Shamir,Adleman により開発された方式、MD5, MD2,SHA-1,SHA,DSA,楕円 DSA,DH
Keymate/Crypto	(株)日立製作所	ELCURVE,ECDSA,ECDH,MULTI2,DES,Triple-DES, Rivest, Shamir, Adleman により開発された方式、EHASH, SHA-1,MD5
SecureWare 開発キット	日本電気(株)	DH, DES, Triplr-DES,RC2, MD2, MD4, MD5, SHA-1, Rivest, Shamir,Adleman により開発された方式
PKCS ｷｯﾄ	NTT ｴﾚｸﾄﾛﾆｸｽ(株)	DES,Triple-DES,RC2 相当,RC5 相当, FEAL, MISTY1, RSA 方式,MD2,MD5,SHA-1

(2) 認証機関構築ツール

代表的な認証機関構築ツールを表 6.2-2 に示す。（順不同）

表6.2-2　認証機関構築ツール

注：各社製品カタログ及びＷｅｂを参照し記述

名　　称	メーカー	特　　徴	プラットホーム
BSAFE Cert-C	RSA Security 社	電子署名:ASN.1 X509,PKCS#10,PKIX に準拠。ｴﾝﾄﾞﾕｰｻﾞ秘密鍵保管に IC ｶｰﾄﾞが使える。	Windows95,98,NT その他 UNIX 系
MistyGuard CERTMANAGE (ｱﾌﾟﾘｹｰｼｮﾝｿﾌﾄ)	三菱電機(株)	X509 Ver.3 に準拠、PKCS, SSL, S/MIME, LDAP に準拠。ｴﾝﾄﾞﾕｰｻﾞ秘密鍵保管に IC ｶｰﾄﾞが使える（ｵﾌﾟｼｮﾝ）	WindowsNT4.0
認証サーバ	(株)日立製作所	PKIX, X509 に準拠、ｴﾝﾄﾞﾕｰｻﾞ秘密鍵保管に IC ｶｰﾄﾞが使える。	HP-UX v10.20
PKI サーバ Carassuit	日本電気(株)	X.509V3 準拠、PKCS＃12 対応、LDAP 連携、IC カード対応可能。一括登録・発行可能。	WindowsNT4.0
プライベート CA ビルダー (S/MIME,WEB)	NTT ｴﾚｸﾄﾛﾆｸｽ(株)	プライベート CA 構築ツール X.509 v.3, PKCS#10, PKCS#7 に準拠。	WindowsNT4.0

(3) その他のツール

SSL, S/MIME, VPN 構築などのツールを表 6.2-3 に示す。（順不同）

表6.2-3　SSL,S/MIME,VPN 構築などのツール

注：各社製品カタログ及びＷｅｂを参照し記述

名　称	メーカー	概　　要	プラットホーム
SSL-C、SSL-J	RSA Security 社	SSL-C はSSL-J 対応アプリケーション開発のためのツール(C 言語、JAVA 言語)	Windows95,98,NT その他 UNIX 系 JAVAﾌﾟﾗｯﾄﾎｰﾑ
S/MIME-C	RSA Security 社	電子メールでのｾｷｭｱﾌﾟﾛﾄｺﾙを実現するための C 言語用ツール	Windows95,98,NT その他 UNIX 系
MistyGuard CryptoSign	三菱電機(株)	電子メールでのｾｷｭｱﾌﾟﾛﾄｺﾙを実現するための C 言語用ツール（S/MIME 準拠）	Windows95,98,NT
MistyGuard TRUSTWEB	三菱電機(株)	MISTY による暗号化とﾃﾞｼﾞﾀﾙ署名によるセキュアな WWW サーバの構築ツール	Windows95,98,NT (ｻｰﾊﾞは WindowsNT)
Secure Socket	(株)日立製作所	MULTI2 による暗号化、X509 準拠の認証による VPN ツール（認証サーバが必要）	Windows95,98,NT (ｻｰﾊﾞは HP-UX 含む)
SOCKSVPN	日本電気(株)	DES による暗号化、公開鍵認証による VPN 構築ツール（SOCKS Ver5 を使用）	Windows95,NT (ｻｰﾊﾞは UNIX 含む)
S/MIME 暗号ｿﾌﾄｳｴｱ標準ｷｯﾄ(開発中)	NTT ｴﾚｸﾄﾛﾆｸｽ(株)	電子メールでのｾｷｭｱﾌﾟﾛﾄｺﾙを実現するための C 言語ツール（S/MIME 準拠）	Windows95,NT

注：SSL や S/MIME､VPN などの詳細については、それぞれ専門書を参照のこと

(4) 暗号ライブラリを使用する場合の、設計時、運用時における留意点

1. 設計時：鍵の生成部や、暗号鍵入力部、パスワード入力部、一方向性関数の出力部、などは、外部への出力はしないようにし、また内部メモリへ一時的な保管をする場合も短時間とさせる。(鍵テーブルのようなものは使用しない)
2. 運用時：パスワードは、第三者に推測され難いものとし、使用有効期間を設ける。
3. また、平文入力や暗号文、復号した平文の保管は、当然のことながらその箇所での盗聴や盗難し難いものとする。

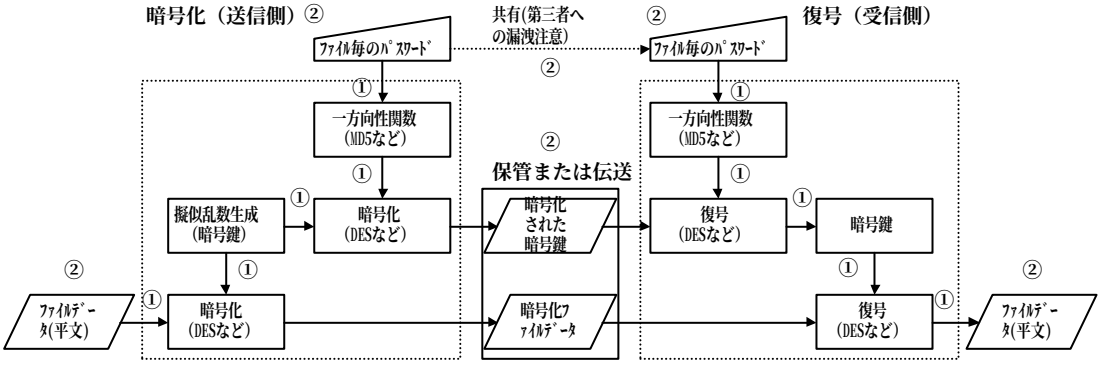


図6.2-1　暗号ライブラリを使用する場合の留意点

6.3 暗号機能の検証

(1) 暗号ライブラリを使用した場合の検証

暗号ライブラリや暗号ツールを使用する場合は、検証用としてメーカーから、平文、鍵、暗号文を何組か提供される場合もあり、暗号機能の検証はそれらで行うのが一般的である、また暗号文は平文と違い特徴的な規則性が見えないため、その保存や通信においては、フォーマットやプロトコル（ヘッダ情報など）の整合性を各機種間で確認する必要がある。

また暗号ライブラリ以外で自ら構築したセキュリテイ上重要な箇所は、第 3 者を交え検証することがよい。（セキュリテイホールとなる可能性があるため）また暗号ツールを使用したアプリケーションの構築では、ユーザからの依頼(別料金)により検証してくれる場合もあるのでそれらを活用するのがよい。

(2) 暗号アルゴリズムの実装開発から行った場合の検証

使用する機種や、OS などの違いにより、動作不良や性能などに大きく影響するので、詳細な検証が必要である、以下 6.3.1 からその留意点などを記述する。

6.3.1 検証の必要性

(1) 複数機種によるシステム構築時の問題

データ交換、トランザクション処理での接続におけるプロトコル、各種ツールとのインタフェース設計に配慮しなければならない。

1. 異なる文化で育ったプラットホーム間での問題メインフレーム、オフィスコンピュータ、ワークステーション、パソコンでのアーキテクチャの違い、データ表現やコード体系の違い。
2. 日本固有の問題として、同一文化圏であってもメーカ毎の差違メインフレーム文化圏であっても富士通・IBM・日立・NES・UNISYS など、多数のメーカー・機種・ＯＳに対応しなければならない場合がある。データ管理における制約の違い（最大レコード長、削除レコードの扱い、etc）、COBOL における演算精度・データ表現などの非互換、高速処理が要求される場合の言語(アセンブラ、HPL：NEC, RPG：AS400, etc）、ネットワークアーキテクチャ（TCP/IP, FNA, SNA, HNA, etc）の違いなどにも配慮する必要がある。パソコンにおいても多様なプラットホームという問題がある。ＭacOS・Windows3.1／95／98／2000／NT などOS の違い、AT 互換機・PC98・Mac などハードウェア環境の違いがある。
3. 相互間で開発時の問題以外にデータ交換における問題、運用における問題があり、検証および保証をどのように行うかが重要である。

(2) 複数開発者による相互運用性の保証

1. 同一プラットホームであっても開発者が異なれば設計思想と実装方法の違いがあり得る。
2. 性能追求かインターフェイスの標準化重視かなど。
3. ユーザインターフェイスおよび機能面での差違があるかの検証。

実装上の差違によりデータ交換・相互接続面で問題がないか比較検証。これら複数

の開発者が開発したシステムの相互接続、および複数開発者が開発した製品やツールを採用したシステムの検証を、各メーカー・ユーザ・第３者機関がどのように分担するかという問題がある。

6.3.2 暗号機能の検証方法

(1) 暗号ロジック実装の検証

通常のシステム検証と同様に、サブシステム単位での検証を行った上で、それらを順次結合しより上位のブロックとしての検証を進めるのが普通である。ここでは暗号化／復号機能を提供するモジュールの暗号ロジック単体の動作確認時に必要な点を挙げる。

① インタフェース仕様の確認

あたりまえのことであるが、どのような単位で暗号化／復号処理を行うかはシステムの要件や使用する暗号方式、モードに左右される。モジュール間の引数の型、長さ、引き渡し方式（スタック利用 or ポインター）など、想定されるデータに対してオーバーフローなどを起こさないことが前提である。通常はブロック単位で暗号化を行うため、パディング処理もどのように行われるべきであるかを確認する（アルゴリズムによりある場合と無い場合がある）。また、引き渡す平文／暗号文／鍵などは、利用後直ちに記憶装置上から消去する必要があるが、どのモジュールが行うかも予め確認が必要であろう。

なお、市販の暗号機能モジュール利用の際は、上記の点がどうなっているかメーカーに確認が必要である。

② 暗号化結果の確認（サンプルテスト、ランダムテスト）

実際に暗号化を行い実行結果を検証する。サンプルテストでは、暗号アルゴリズムといっしょに提供される、特定の平文／鍵／暗号文の組み合わせが実現できるかどうかを確認する。このテストでは、実装結果が暗号アルゴリズム上正しいかどうかが判断できる。

あるいは、すでにある製品やモジュールとの整合性をとるのであれば、それらの製品へ適当なサンプルデータを入力した結果と比較することも良い。

ランダムテストでは、とりうる範囲のできるだけ多くの平文と鍵をランダムに選択し、これにより出力される暗号文を統計的に検査し、分布の偏りや bit 毎の０/１の割合などを検査する。これにより、実装された暗号アルゴリズムと、入力される平文が妥当な範囲であるかどうか判断できる。極端な偏りが発生するような場合は、その平文の集合はそのアルゴリズムを使用するのには向かないこととなるので、平文に対する何らかの処理（冗長データの付加、圧縮など）を行う方が良い（通常はアルゴリズムを変更することができないため）。

なお、公開鍵暗号方式では鍵の生成に時間がかかる上､暗号結果のばらつきは公開鍵を実現する数学的性質を考慮する必要があるため、通常このテストは行わない。

③ 復号機能の確認

上記と同様に（通常は同時に）、実際に復号を行い、正しい結果が得られることを確認する。

④ 相互の暗号化／復号テスト

通信やメディアの受け渡しが発生する場合、実際の通信やメディアの交換を行い、他の機器やソフトウェアで暗号化されたものが正しく復号できることも確認する。

特に、6.3.1 の機種毎の差異についての確認が重要な点であり、データ列の並びに気をつけなければならない。

⑤ セキュリティ上の欠陥チェック

暗号モジュールは解析されたり不正な改造がなされないように、厳重に管理される必要がある。実行モジュール内部でも不正な改竄が無いかどうかをチェックする機能は不可欠で、これらは実際の解析／攻撃を行ってみなければ確認できない。

また、モジュール内およびモジュール間で使用される変数領域も、不正に監視されないようプロテクトされあるいは解読できないことを確認しなければならない。

(2) 鍵管理システム実装と運用の検証

鍵管理システムにおいても機能検証は欠かせない。特に鍵交換・鍵配布については、相互のやり取りが発生するために、機種毎の差異があると問題となるので十分な検証が必要である。

その他、鍵の生成（および更新）時には、正しく鍵がランダムに生成されているかも確認しなければならない。また、鍵の登録／廃棄については、一種のデータベースシステムとなるので十分なトランザクション能力と長時間耐久性なども検証しなければならない。

最後に、鍵管理システムとしてもっとも重要なのがセキュリティである。セキュリティに関しては、実際に攻撃を行い、問題が発生しないことを証明するだけではなく、実運用時にあらかじめ想定/検証した以外の手口で攻撃されないかどうか、また運用上のログや記録は正しく安全に保管されそのチェックと対策がなされているかなど、運用上の監査が必要である。セキュリティに関しても通常のビジネスプロセスと同様に、PLAN-DO-CHECK-ACTION というサイクルが重要であり、後２者が欠けていては意味を成さないことを十分にご注意いただきたい。

(3) 標準化されている検証方法

JIS や ISO のような機関では様々な仕様は定義するが、実際の検証方法や相互運用性確保のための検証方法などは定義しないのが普通である。あくまで、暗号開発者の提供する検証データにより確認することが求められる。ただし、米国においては NIST が DES の実装と利用に関するガイドライン[1]を規定している。これには実装方法と鍵管理、キャラクタセットとの対応などが述べられている。

また、一般的な暗号モジュール（ここでは広くハードウェアや鍵管理、OS なども含める）の検査プログラムも規定されている[2]ので、参考としては使用できる（ただし、DES，Skipjack，DSA，SHA－１だけである）。

(4) 暗号実装者による検証

実装者による検証は、当然必要である。この場合も、できるだけ検証方法を体系付けて定義し、それに則った検証作業が必要である。また、検証結果については、不正な改竄がされないよう正しい手続きで安全に保存する必要がある。

(5) 第３者による検証

実装者と利用者だけでなく、第３者が客観的な検証作業を行うことも必要である。

現在はそのような組織や企業はほとんど見られないが、今後はこのようなこともビジネスとして取り組むところがあるかもしれない。また、暗号機能に関する保険的考え方をすれば、保険機関などがこのような業務を行うかもしれない。

いずれにしろ、利用者にとって評価基準が広く統一できることは良いことであるが、検証手続きの正当性や、コストと内容などサービスの対比など、ユーザの考慮すべき点が増えることも注意が必要である。

6.4 暗号利用とそのコスト

暗号を利用する場合は、しない場合に比べて当然コストが増大する。このコストはセキュリティを守るために必要なコストであるが、実際のシステム構築においては、算出することが困難であったり、想定されないコストが発生することもある。

一方、安全性とコストの比較でいえば、10 万円の価値の情報を守るために 10 万円のコストをかけることは、自由競争下の企業においてはナンセンスであるが、どのような情報がどれだけの価値を持っているかを実際に検討している場合は少ないと思われる。

ここでは、暗号利用に必要とされるコスト要因を検討し、リスクとコストの考え方を提示することにより、暗号利用システムの設計・構築にあたる諸氏の参考になればと考える。

6.4.1 暗号利用の問題点

(1) 原理的問題点

暗号を利用する場合の大きな問題点として、一度導入した暗号技術は、常に進歩する解読技術と計算機能力の向上により、安全性の低下が避けられないことがあげられる。一方では、解読技術の進歩が新たに強力な暗号技術を生み出すというジレンマもあり、解読技術の向上を止めるわけにもいかない。このことから、暗号利用に際してはそのシステムの寿命を考慮しその期間中十分な安全性を保てるような対策が必要である。

最も簡単な手法としては、十分に長い鍵を使用することである。2.2.4 でも述べたように、計算機能力の向上やアルゴリズムの高速化を考慮しても、100bit 以上の共通鍵暗号を容易に解読するのは、今後十数年は困難であると考えられている。同様の手法としては、鍵長が可変の暗号アルゴリズムを使用し、時間とともに鍵長を増大させていくことが考えられるが、システム構築コストとしてはたいして変わらない上、鍵長を判別する手法や、鍵長を変化させた際の検証など複雑化する要因が多くなる点に注意が必要である。よほどの理由が無い限りは、はじめから十分に長い鍵を使用する方が望ましいと考えられる。ただし、どれだけ長い鍵を用いても、安全性は「確率的に」保証される点は注意が必要である。

すなわち、総当たり法で次々と鍵を試行していく場合、運が良ければ最初に鍵が見付かる確率もゼロではない。一方最後の鍵が正しい暗号化鍵である可能性もあり、平均として全ての鍵の半数を試行すると 50%の確率で解読できるに過ぎない。

別の手法としては、暗号アルゴリズムを複数利用できるようにすることである。す

なわち、ある暗号アルゴリズムの安全性が極端に低下した場合、直ちに別の暗号アルゴリズムへの移行ができるようなシステムを構築することである。理想的には、常に複数のアルゴリズムが選択可能であることが望ましいが、後述するようなロイヤリティや工業所有権にる費用が増大する、使用されているアルゴリズムを特定するための管理手法が必要であるなど、コスト要因もある。また、アルゴリズム毎に鍵の選択基準（弱い鍵などの排除）が異なるため、単に暗号化／復号処理だけでなく、鍵生成・更新処理も対応が必要である。

仮に、様々なアルゴリズムが選択可能であったとしても、相互運用性を保証するには少なくとも１つの共通なアルゴリズムについては利用可能でなければならない。

実際、DES が非常に普及した背景には標準的手法を持たないものが淘汰されるという市場選択原理が働いたともいえる。しかしながら、このように単一のアルゴリズムにシステム全体が依存してしまうと、そのアルゴリズムの欠陥が判明した際には、対策が膨大な量になるとともに、特に電子商取引など信用が重視される場合においてはシステムに対する不信感が、場合によっては取引きの停止など社会的に大きなダメージを与える恐れがある。

従って、暗号技術の実装においては複数アルゴリズムを利用可能なようにシステムを設計構築し、できる限り複数の標準手法をサポートすることが望ましいといえる。

余談であるが、AES において複数候補が標準になる可能性を否定しないのも同様と思われる。一方 ISO では暗号アルゴリズムの登録制度を一歩進めて標準化させようという動きもある。

(2) 実装上の問題点

暗号機能を実装する場合は、個々のアルゴリズムの安全性よりもシステムとしての安全性に対する配慮の方が重要である。どんなに優秀なアルゴリズムもその価値を十分に引き出す鍵の生成、配送、保管システムが無ければ、絵に描いた餅にすぎない。

最適な暗号アルゴリズムは、解読のための時間や手間、コストが、他の手段で元の情報を入手するよりも、多くなることを提供できればよいのである[1]。

特に、システム全体への攻撃の想定は非常に重要であり、どの部分が弱点になるかを予め判断しながらシステム設計を行う必要がある。たとえば、IC カードなどでは、物理的所有者と情報の所有者が、同一の場合と異なる場合で、攻撃者の攻撃内容が異なるからである。

すなわち、PC 内のデータのように個人が必要とする情報を個人が保持する場合は、第３者のアクセスを防止するという物理的なセキュリティが確保されれば、比較的安全といえる。

一方、他者の情報を保持する場合には、物理的・論理的なアクセスが不可欠である。

たとえば、電子商取引におけるサーバや、認証機関のサーバ、不特定多数の操作が前提となる端末機器などは、通常そのシステムの利用者の情報を保管することとなる。IC カード型電子マネーやネット上の e キャッシュに格納されている価値（バリュー）はもともと発行元の物であると同時に、特定の所有者に限定されない転々流通性を持つ物であり、所有者を問わず同じ価値を持つものとなる。

これらの中に保管されている情報は通常、強固なトランザクション・セキュリティ

や耐タンパー性のメディアにより保護されるが、逆に言えばこれらにより防止しなければ、情報や価値を自分の都合の良いように改竄・偽造することで利益を得ようと、誰もが考えるということである。したがって、このような場合は情報を格納している個別要素そのものの安全性を高めると同時に、改竄や偽造を検知し他のシステムに影響をおぼさない仕組みが必要である。

鍵や秘密情報を守るための最上の方法は、物理的な（静電・電磁気的結合を含む）接触そのものを完全に遮断することである。たとえば、認証システムで使用される OTP（使い捨てパスワード）生成パッドなどは、物理的に認証システムとはつながれず、利用者の目視と手入力が介在するし、計算機の CPU と OS を使用しない認証システム（認証機能付きキーボードやキーボード・インタフェースの認証装置）は、万一攻撃者が特殊なソフトウェアを仕込んでも、PIN を盗聴することは不可能である。

また、実装時に特に注意が必要な点としては、ソフトウェア上のエラー処理やプロセッサのエラー処理などの例外処理のパターンが解析されることにより、暗号の解読が容易になることがある。実際、電磁波放射による IC カードの誤動作パターンによる攻撃事例がある。

例外処理は、アルゴリズムの規定には当然記述されてないので、実装者がシステムにあわせてどのような処理を行うべきか判断する必要がある。

実際のシステムの実装時には、どのような点に留意すべきかを以下に例として述べるので、上で説明した各項目を考慮しながら、各自で検討を行っていただきたい。

① 秘密情報の暗号化

本人認証の為の情報、プライベートキーファイル、プライベート ID ファイルなどの秘密情報は他人に不正使用されないように暗号化されていることが望ましい。一般的にはパスワードなどを用いて暗号化する。

② 秘密情報の削除

パスワードなどで暗号化された秘密情報は、パスワードを全数検索（全数探索、総当り）する事によって暴露される可能性がある。従って全数検索をさせないように、一定回数以上のパスワード不一致が生じた場合、秘密情報を削除などによって使用不能にすることが望ましい。

③ 秘密情報のコピー防止

上記のような使用不能処理をしても、秘密情報のコピーが容易に得られては効果が無い（使用不能になっても復帰が可能）。従って、コピーされた秘密情報は正常に動作しないようにする事が望ましい。

④ 秘密情報の格納場所

一般的にフロッピィディスク（FD）は複製が得られやすい（いわゆるデッドコピー）。秘密情報を FD に格納した場合はコピー防止機能が充分に効果を発揮しない場合がある。

また、ネットワーク上に格納する場合はアクセスを適切に管理しなければコピーされる場合がある。従って、これらの場所に秘密情報を格納する場合は、その安全性、運用を慎重に検討する必要がある。

⑤ 本人認証の機能

一般的にパスワードの全数検索には、値を順次に変えて入力するプログラム（パ

スワードクラッカーと呼ばれる）を用いて自動的に行う。これはパスワード検証（パスワードの一致、不一致を検証する）のみを行う関数（機能）が用意されている場合に有効となる。

従ってパスワード検証機能は単独で機能するのではなく、暗復号等の処理と一体化しており、本人認証が必要になった場合にのみ機能する方が良い。また、パスワードは関数のパラメータとして与えるのではなく、ダイアローグを開いてユーザが直接入力するなどのリダイレクトできない入力方法を取ると、全数検索の自動化がし難くなる（＊デバッガなどを使用すれば必ずしもリダイレクトできない訳ではないが、その実行には非常に時間がかかる）。

⑥ 本人認証情報入力時のカモフラージュ

パスワード入力は背後から覗かれる危険が常に伴うが、入力されたデータをそのまま使用するのではなく、前方一致や後方一致などの方法を取り込めば、入力時に余分なデータを入れ込むことでカモフラージュとなる。

⑦ 処理速度の均一化

例えばデータベースからデータを検索する場合、前半に当該データが存在していれば検索時間は短いが、後半に存在していれば長くかかる。これによりデータベース中のデータの分布が判る。同様に、ある関数の処理時間がその関数に与えるパラメータと相関がある場合、予期せぬ情報が漏洩する場合がある。これを防ぐ為に時間調節の為のダミー処理や、パラメータそのものの均一化（値の幅、bit の分布）を計る必要がある。

⑧ 関数の深階層化

実行オブジェクトから逆アセンブルを経てプログラムを解析する場合、幾重にも関数コールが存在すると解析者の負担は非常に増大する。階層のレベルによって独立オブジェクトとして存在していると、構造を理解する手助けになるので、全ての階層が一つにまとまっている方が解析の精神的な妨げになる。

情報のカプセル化（クラスなど）、消去情報のカプセル化を行い、その存在を隠す。

その他に、使用したメモリは全てクリア（またはダミーデータの書込み）する。また、プログラム終了後に秘密データの痕跡が残らないようにする。

⑨ 実行モジュールの暗号化

多くの場合、実行モジュールを解析するにはまず、実行モジュールのファイルを逆アセンブラで処理し、解析ファイルを紙に印刷し机上でトレースを行う。実行モジュール（の本体）が暗号化されていればこの処理がおこなず、デバッガを用いて少しずつプログラムを走らせ、メモリに展開されてからファイルに出力させなければならないので手間がかかる。

6.4.2 リスク分析

そもそもコストをかけて暗号化を施すデータは、そのデータが解読されたりした場合にどのような影響を及すかのリスクを算出したうえで、必要となる対策を施さなければならない。

ここでは、おのおののデータが持つリスクとして何を考えなければならないかを検討す

る。

(1) 解読（漏洩）リスク

第一に､暗号化データが解読された場合、どのような影響があるかを検討しなければならない。この場合の解読は、鍵を特定できるレベルの解読と、鍵は特定できないが暗号文に対応する平文を求めることができるレベルに分類できる。

鍵が特定される場合は、解読されたデータだけでなく同一の暗号化鍵（共通鍵、公開鍵）で暗号化されたデータについても復号が可能となるので、全体としてのリスクを考慮する必要がある。また、鍵の特定無しに暗号文から平文が解読されるリスクについては、通常特定の平文と暗号文のペアがすでに入手されていることや、同一の鍵による複数の暗号文が必要になるため、セクション鍵の更新頻度を適切にすることで原理的に困難とすることが可能である。一般的には秘密情報などが読み出される場合にどのような影響があるかを考慮する必要があり、秘密情報自体の価値の算出と、波及する影響の総和を検討しなければならない。具体的には情報の作成、入手に関わる費用、情報を第３者が入手し使用した場合の損失、契約書類など相手がある場合の補償、プライバシーなど判例から想定できる補償などが対象となる。

(2) 改竄リスク

上記のうち、鍵が特定されてしまう場合には、改竄、偽造という操作が可能になる。

そのうち、改竄については、鍵を特定されなければ（確率的には）一般的に不可能なレベルであるといえる。改竄のリスクは、本来改変されないはずの情報が改竄（変形）させられることにより、誤った情報を伝達することによるリスクである（保管も時間軸の離れた伝達といえる）。

ただし､改竄は本来伝達されるべき内容のごく一部について行われることが多く（金額の改変や口座番号の改変など）、アプリケーション上で内容の改変を特定できるしくみが無ければ検出が困難である。この場合のリスク算定は、条件付けにより大きく変動する。

たとえばクレジット決済のように上限を設けても、どの程度の回数で改竄が行われるかにより、被害が数桁変わることもまれではない（テレホンカードやパチンコカードなどの例）。

(3) 偽造リスク

偽造は、一見困難なようであるが、鍵が特定できずに解読が可能なレベルであれば、複数の暗号文から別の暗号文を偽造することが可能な場合がある。

ただし、偽造の場合はたとえば元の平文が存在しない状態で新たな暗号文が生成されるので、通信の場合にはシーケンス番号が重複する別の暗号文が存在したり、保管の場合にはもとの情報やメディアを消去しない限り２つのもの（情報）が存在するという矛盾から、検出されることが用意である。このように、アプリケーション上で対策を施すことが用意である点、再犯が困難である点などを考慮すると、改竄ほどシビアな算定は必要無いともいえる。

(4) なりすましリスク

なりすましとは、広義には認証手段をごまかして、別の対象（エンティティ）になりすますことを言う。一般には別人になりすまして秘密の情報へアクセスしたり、架空の取引きを行うなど、情報の漏洩や損害の発生（この場合他の利用者）がつきものである。これらについては、個々に上記のようなリスクを算出しなければならい。

暗号技術を利用しない認証（暗証番号やバイオメトリクスなど）の場合は、認証技術としてのリスク管理（誤認証や認証システム自体のトラブル対策）はもちろん必要である。

ただし、本人認証のためのシステムであるため、認証システムを更に人間がチェックするのであれば、なりすましのリスクは小さいといえる。

一方、2.1.2.1 のような電子署名を用いた認証や、UNIX のパスワードや CHAP(Challenge Handshake Authentication Protocol)など暗号技術を利用した認証の場合、前述のような解読・改竄・偽造などの手段によりなりすまされた場合は、通常はそのメッセージだけが到着するため、本人かどうかを別の手段で確認することは不可能に近い。従って、なりすましそのものの回数がどの程度まで許容されるかも考慮して、リスクを算定しなければならない。

(5) 鍵遺失リスク

一般にマスター鍵など重要な情報は厳重に保管するが、情報の性質上無闇にバックアップをたくさん取ったり可読性の有る媒体（メモなど）にして残すことはセキュリティ上問題である。

一方、計算機上で保持する情報は、地震や火事などの災害や機器の故障、ソフトウェアの不具合、操作ミスや人為的な破壊行為により失われる可能性が常に存在する。したがって、共通鍵暗号による情報（データ）の保管や、公開鍵暗号による電子署名と暗号化を行ったものを保管する場合は、鍵が失われた場合にどうなるかを予め検討しておく必要がある。

6.4.3 コスト要因

(1) 暗号処理コスト

暗号機能をシステムに付加する場合は開発・実装費用が必ず必要になる。これには設計レベルから検証の費用などに加え、通常の処理部分においても、暗号化機能を付加する構成要素との連携のために必要な設計上の配慮や、検証作業が困難になる点など通常のシステム以上の費用が必要である。また、暗号機能部分を外部から調達する場合は、調達のコストも必要である。

また、一般に暗号は特許対象になることも多く、ロイヤリティ不用の場合を除き、知的所有権に対する費用が発生する。外部から調達した場合などは、モジュールのインストールベースで課金されることもあり、調達条件と含めて考慮が必要である。

以上はソフトウェアや人的資源などに関するコストであるが、むしろ問題となるのは暗号化・復号のために必要な資源（の増大）である。通常の処理以外に暗号化・復号を行うために、処理プロセッサを強力なものにしたり CPU を増やすなどの対応が必要となる。また、暗号化処理はかなり時間が必要であり、応答時間に対する要求がシ

ビアな場合は、その分処理能力を上げるか、専用回路などの高速化の手法を導入せねばならず、これらのコストもあらかじめ想定しなければならない。

また、暗号データは圧縮が困難であるため、通信経路のスループット低下やバックアップ媒体の容量不足などを招かないよう、すべて非圧縮状態での算出が必要となる（一般的には２～３倍程度の容量を必要とする）。

更に、システムを稼動させるにあたっては障害対策や障害の切り分けのための運用・管理ツールが不可欠であるが、暗号化機能を付加したシステムでは暗号化されたデータの処理経路（トランザクションや通信経路）上でのトラブル時に対応できるようなツールの作成時にも、暗号化や復号の機能や、トレース情報の比較など、通常の処理システムに比べて設計・製作・検証に手間がかかる点も見落としてはならない。

最後に、どのような暗号化機能であっても、一般的なコンピュータのハードウェア、OS 環境下で処理を行う以上、暗号化・復号機能モジュールや作業データなどを安全な条件で操作できるよう、特別な配慮が必要である。このことはシステム設計レベルから運用レベルに至るまで、全ての処理を通じて安全性に対する配慮とロック機構が必要となることを意味し、実現には十分なノウハウと厳しい運用管理が求められる。このようなコストは開発のみならずシステムのライフサイクル全体にわたり影響するため特に重要である。

(2) 鍵管理コスト

実際の暗号機能の実現には鍵の管理コストが不可欠となる。安全な鍵を生成するために必要な資源は、更新も含め、常に必要なコストである。また、公開鍵は生成にかなりの時間と資源を要するため、数多く生成する場合はこれらについて十分検討を行う必要がある。

上に述べたように、鍵が失われた場合に情報が復元できなくなるリスクを回避するために、鍵を安全に保管し、なおかつ失われた鍵が必要な時に迅速に鍵を回復するための鍵管理システムを用意する場合は、これらの構築と運用･管理のためのコストも必要となる。また、鍵の交換を行うためには、双方が鍵交換するための処理が必要であり、通信媒体上でもそのようなトラフィックが発生するので、これらも考慮に入れる必要がある。

そして、鍵を安全に保管することはほとんどの場合必要である。この場合は、どのような手法でどのようなメディアを使用するかにより変化する（もっとも安価で安全な方法は人間が記憶することである）。なお、公開鍵暗号系を使用する場合は、廃棄鍵管理の費用や認証機関（内部認証機関）の構築・維持コストなども考慮に入れる必要がある。

(3) 外部機関による保証のコスト

電子商取引など、金銭に関わる暗号利用に関しては、クレジットカード決済のように事故などに対する補償として保険などの仕組みの導入が必要である。このようなコストも運用コストとして外部に費用支払いが発生する。

このような場合、システムの機能が十分であるか、また安全性が問題無いレベルであるかどうかを第３者の立場で検証してもらう必要がある（保険機関などが行う場合

はその費用に含まれる）。また、外部認証機関を利用する場合はこれに対する費用算出が必要であるが、サービス事業者が少なく選択の余地が乏しい、実績が少なく価格変動のリスクがあるなど問題も多い。今後は国内事業者の増加と、サービス品質などのメニュー構成の充実で、ある一定の相場が成立することを望みたい。

7 暗号に関する制度・法令

7.1 暗号適用時の留意点

暗号技術は、主に軍事や外交の分野において利用され発展してきた経緯があるため、暗号の利用／適用や輸出入について、一般の製品とは異なる規制が実施されている。このような規制があるため、暗号システム構築や、暗号利用においては、十分に注意が必要である。注意点の具体例を以下に説明する。

(1) 違法コピーによる利用

違法コピーによる利用で、使用者特定で輸入した暗号組み込みソフトウェアを違法コピーし使用した場合、輸出入業者が罰せられる可能性もあるので、注意する必要がある。

(2) 自動配付システムの利用に関する問題

米国の企業や研究所におけるサイトから自動配布システムにより暗号ソフトを入手して利用することが可能なケースがあるが、上記のような各種法律に抵触することがあるので注意をする必要がある。また、日本からの輸出の場合にも米国製の暗号が入っている場合には同様に米国の法律に抵触することになり、注意が必要である。

(3) 利用暗号の特定の問題

また、SI 事業者が輸出するソフトウェアに、使用されている暗号と鍵長を明示する必要がある。

7.2 暗号の位置付けと政策問題

7.2.1 暗号に関する政策の項目

(1) 暗号の利用と貿易に関する政策

暗号技術は、主に軍事や外交の分野において発展してきたという歴史的経緯があるため、その輸出に一定の規制をおぼしている国が多い。また、1996 年に発足した武器輸出管理のための国際的な枠組みである、ワッセナーアレンジメント（WA：Wassenaar Arrangement）においては、暗号機器は通常兵器などと共に輸出管理品目リストに上げられている。WA 参加各国はこのリストに基づいて、個別に輸出管理を行っている。

最近の 1998 年 12 月に WA の改定が行われ、一定の強度（DES56bit 相当）を超える暗号を管理対象とし、これ以外の暗号製品については輸出管理の対象外とした。暗号技術の先進国である米国においても輸出規制の緩和の方向にあり、今後の動向を見ていくことが重要である。

なお、暗号技術の国内使用については制限していない国がほとんどである。

(2) 認証に関する政策

公開鍵方式の暗号技術が人々の信頼の下に円滑・効果的に利用されるためには、通信の相手方の真性な公開鍵を確実に入手できることが前提になる。そこで、公開鍵の真性性を証明しつつ公開鍵の管理・配信などの業務を行う、認証機関（CA：Certification

Authority）の存在が必要になる。

特に、公開鍵の登録において本人確認が厳密に行われなかったり、公開鍵の管理・配信の在り方がずさんな場合には、社会秩序が混乱するばかりでなく、なりすまし犯罪やマネーロンダリングなどの蔓延を助長する恐れもある。そこで、認証機関や認証の在り方については、各国や各種国際機関において各種の検討が行われている。詳細については、ECOM 認証局検討 WG の調査報告書（認証に関わる諸外国の法制度調査報告書）に記載されている。

(3) キーリカバリ（鍵回復）／キーエスクロー（鍵預託）に関する政策

キーリカバリ（鍵回復）機能を確保しておくことは、電子商取引を始めとする様々な場面において、鍵の紛失や事故などによる消失の際のトラブルを解決するために大変重要である。しかしながら、このキーリカバリ機能については、個人のプライバシーとの整合性に係る問題点も指摘されており、この点についても配慮しながら検討を進める必要がある。

暗号の不正利用による犯罪が発生した場合に実効的な対処手段を確保するためには、暗号化されたデータへの合法的なアクセスを可能にしておく、キーエスクロー（鍵預託）の制度が米国などの幾つかの国で提案された。

7.2.2 国際社会での議論の動向

(1) G7／P8 テロ関係閣僚会議

1996 年７月に開催された G7／P8 のテロ関係閣僚会合で、テロリズム対策に関する合意文書を発表し、この中で、暗号について以下のように述べている。

「合法的な通信のプライバシーを保護しつつ、テロ行為の抑止・捜査のために必要な場合に、政府によるデータおよび通信への合法的アクセスを可能にする暗号技術の使用に関する協議を、適当な二国間または多国間のフォーラムにおいて促進することを全ての国に要求する。

(2) OECD 暗号ガイドライン

OECD（経済協力開発機構）は 1995 年 12 月に、最初の会議「暗号方針に関する OECD 専門家会議」を開催した。この会議で、国際的な協調が必要であること、またプライバシー保護と公共安全のための「法に基づく入手」のバランスに関して統一的な解決策が必要であることが確認された。暗号機能を適用するためのガイドラインの最初の原案は、国際商工会議所と OECD の民間諮問委員会である BIAC が作成した。

最終案は 1997 年３月に OECD 委員会で承認され「OECD 暗号ガイドライン」として公開された。このガイドラインは、８原則からなり、プライバシー保護との整合性をとりつつも、暗号化されたデータを国家などの第３者が強制的に解読することを認めたものである。

(3) EU（欧州連合）

欧州連合閣僚理事会においては、1995 年９月、加盟国に対する「情報技術に関連する刑事訴訟法の問題に関する」勧告を採択し、その中で、暗号の使用に関して次のよ

うな内容の勧告を行った。「暗号の合法的な使用への（抑止的な）影響を最小限にとどめつつ、暗号の（不正）利用が犯罪捜査におぼす否定的な影響を最小にするための手段を考えるべきである。

また、欧州委員会（Europe Commission）においては、ETS（Europe-wide Network of Trusted Third Parties Services：ヨーロッパに地域おける TTP サービスのネットワークシステム）の調査研究、およびデジタル署名の法的側面に関する調査研究をそれぞれ約１年をかけて進めている。

(4) ISO（国際標準化機構）

技術の標準化について、ISO／IEC JTC1／SC27（セキュリティ技術）において議論が行われており、TTP（Trusted Third Party：信頼される第３者機関）、暗号技術、セキュリティ評価について、それぞれ WG1，WG2，WG3 において検討されている。また暗号アルゴリズムの登録制度があり、多数の暗号が登録されている。

(5) ITU（国際電気通信連合）

技術的側面について、電気通信に係る国際標準化を行う国連機関である ITU において検討がすすめられている。ここでは、認証に係る標準フォーマットを提供しており、デジタル署名技術の標準としては ITU／ISO／IEC の「X.509 勧告」がある。

7.3 OECD 暗号ガイドラインと注意点

7.3.1 OECD 暗号ガイドラインの８原則

(1) 暗号手法に対する信頼

暗号手法は、利用者が情報システムや通信システムに適用する際に秘匿性が確保できるように、信頼性の高いものでなければならない。

(2) 暗号手法の選択

利用者は適用される法律のもとで、利用する暗号手法を自由に選択できなければならない。

(3) 市場主導の暗号手法の開発

暗号手法の開発は個々の利用者や産業界や政府からの要請にこたえるものでなければならない。

(4) 暗号手法に関する諸標準

暗号手法に関する技術標準や基準やプロトコルは国家および国際的レベルで開発その適用を推進していかなければならない。

(5) プライバシーおよび個人データ保護

通信の秘密や個人データの保護を含むプライバシーに関する個人の基本的権利は国家の暗号方針の策定や暗号手法の導入と運用において十分尊重されなければならない。

(6) 合法的アクセス

国家は暗号政策により、暗号化されたデータの平文、または復号のための鍵への合法的なアクセスを認めることができる。この政策実施にあたっては、本ガイドラインの他の原則を十分配慮しなければならない。

(7) 責　　任

暗号サービスを提供するか、鍵を保管または利用する個人または組織に関して契約書が作成される場合は、その責任について契約書または規則に明記しなければならない。

(8) 国際協力

政府は暗号政策を遂行するために国際協力をしなければならない。この一環として、政府は貿易を阻害するような暗号政策を制定してはならないし、そうした暗号政策が存在する場合には、除去しなければならない。

7.3.2 法に基づく復号の問題点と動向

(1) 問題点と動向

暗号の不正利用による犯罪に対処し公共の安全を保障するために、実効的な対処手段を確保するためには、「法に基づく入手」が必要となる。また一方、一般消費者を含む利用者が安全だと確信できるセキュリティ対策やプライバシー保護対策が益々重要となってくる。これらの二つのバランスに関して統一的な解決策が必要であることが確認された意義が大きい。今後、各国においてガイドラインに沿った各種制度が整備されていくものと思われる。

ただ、具体化にあたっては「法に基づく入手」での情報を保管する鍵管理機関の運用には、監視などを含めて十分な注意が必要である。

(2) その他の問題点

企業などの組織が分割・合併・倒産などになった場合、署名の証跡の管理は難しい問題である。例えば、ある企業の一つの事業部門のみが他の会社に分割・譲渡された場合、新しい会社で過去の事業経過をトレースするために過去の証跡を確かめることが必要である。この証跡の確かめが可能となるためには、署名のための鍵管理が新しい会社に移管できることが必要である。このための法的な環境の整備も必要と思われる。

7.3.3 実装担当者が留意すべき点

実装担当者は OECD 暗号ガイドラインの(1)暗号機能の信頼性、(2)暗号機能の自由選択、(3)市場の要求に基づく暗号機能の開発、(4)暗号機能の標準、の４項目それぞれについて、定量的で具体的な実現をはかることが求められている。

また実装担当者は、OECD 暗号ガイドライン項目(7)責務にあるように、鍵の管理・利用についての責務を契約書に明記することが求められる。

7.4 各国の政策

7.4.1 米　国

(1) 輸出・輸入・使用についての政策の現状

米国においては主として国家安全保障上の目的から、暗号機器の輸出について規制が行われており、ハイグレードの暗号技術（鍵長 40bit 以上）については、国務省の管轄下で原則として輸出が禁止されていたが、最近では以下のような規制緩和された内容となっている。

① DES56bit 相当以下の暗号については、テロリスト支援７ヶ国を除いて輸出が可能である。なお、鍵の配送のために用いられる場合には、共通鍵暗号は 112bit まで、公開鍵暗号は 1,024bit まで輸出が可能である。

② DES56bit 相当を超える暗号についても、以下の用途／仕組みのものは、①と同様にテロリスト支援 7 ヶ国を除いて輸出が可能である。

・キーリカバリー／キーエスクロー製品
・金融関連トランズアクション専用製品
・米国企業子会社

③ DES56bit 相当を超える暗号で、以下の用途／仕組みのものは、指定 44 カ国に輸出が可能である。

・金融機関向け汎用暗号製品
・オンラインマーチャント向け
・ヘルスメディカル・エンドユーザ向け
・キーリカバリー機能付き（企業向け）

④ 上記の①～③以外のものについては、個別審査となる。

(2) 政策の今後の動向

さらに高度暗号製品の大幅な輸出規制緩和を 2000 年 1 月発表し、幾つかの輸出が自由化された。

7.4.2 欧　州

(1) 英国

① 輸出・輸入・使用についての政策の現状

輸出については厳しい規制がある。法執行側のアクセス手段として、信頼できる第三者機関に鍵を預託する TTP（Trusted Third Party）の導入を検討している。

② 政策の今後の動向

あらゆる市販暗号製品は解読可能とするか、またはキーエスクロー扱いとする。個人の使用についてはキーエスクローを義務付ける。

(2) フランス

① 輸出・輸入・使用についての政策の現状

ここ 1 年で暗号政策は大きく変化し、暗号の輸入や利用は自由となり、暗号利用時の鍵登録や鍵の TTP も無くなる方向である。

② 政策の今後の動向

政府の暗号政策に沿って、法律等の環境が整備されつつある。

(3) オランダ

① 輸出・輸入・使用についての政策の現状

個人的な通信に使う暗号の利用は自由である。ファイル暗号化できるものについては認可が必要である。

② 政策の今後の動向

現状維持である。

(4) ドイツ

① 輸出・輸入・使用についての政策の現状

輸出にたいしては緩やかな規制である。個人暗号ソフトウェアの使用は許されている。なお、EU 諸国においては、ECR（EU Council Regulation）や ECR に基づく The Decision などの EU における取り決めをベースとした規制態様があり、具体的には、EU 域外への暗号機器の輸出に限り規制されていることが多い。

② 政策の今後の動向

イギリス、アメリカ、フランスよりはも自由である。ドイツは商業暗号市場に注目している。

(5) イタリア

① 輸出・輸入・使用についての政策の現状

財務省が暗号化の記録にアクセスできるよう義務付けている法律が唯一の暗号法である。

② 政策の今後の動向

郵政省のキーエスクローまたは TTP がある。

(6) ロシア

① 輸出・輸入・使用についての政策の現状

厳しい輸出入法がある。国の許可なしに暗号を開発、製造、設置することは禁じられている。

② 政策の今後の動向

個人に対しては厳しい政策を継続するものとみられる。産業界に関しては、キーエスクローを採用すれば、やや自由化の方向でる。

7.4.3 アジア、他

(1) イスラエル

① 輸出・輸入・使用についての政策の現状

輸出規制ある。しかし、鍵長についての制限はない。強い暗号の使用については軍からの許可が必要など、暗号についての規制はあるが、その範囲は明確でない。

(2) シンガポール

① 輸出・輸入・使用についての政策の現状

輸出規制はない。輸入規制はあるようである。但し、ハードウェアの暗号装置については許可が必要のようである。最近、規制法が施行され、暗号製品の輸出と輸入については、商務省に届け出ることが義務付けられた。暗号の利用については自由である。

(3) 韓国

① 輸出・輸入・使用についての政策の現状

暗号装置の輸入は禁止している。暗号製品の輸出はワッセナーアレンジメントに則って、規制が行われている。

(4) マレーシア

① 輸出・輸入・使用についての政策の現状

輸出、輸入ともに規制がない。最近デジタル署名法の成立し、この法律は TTP への鍵預託を指示しているが、実施手順の詳細はよくわからない。

7.5 日本における制度・法令

7.5.1 暗号製品輸出入に関わる各種法令

我が国においても、暗号機器に関する輸出規制が行われてきた。具体的には、「外国為替および外国貿易管理法」、「輸出貿易管理令」および「外国為替令」などの関係法令に基づき、「暗号装置またはその部分品」などを輸出しようとする者は、通商産業大臣の許可を受けなければならないこととされている。一方、暗号機器の使用および輸入については、法律上なんら制限されていない。

日本の暗号輸出規制はワッセナーアレンジメント（WA）が基本になっているが、最近（1998 年 12 月）WAの改定に伴い、上記の省令・法令が改正された。具体的には、DES56bit 以下、RSA512bit 以下、楕円曲線 112bit 以下の暗号については、輸出規制の対象外となった。また、これらを超える暗号については、許可取得の申請をする必要があるが、包括許可制度などにより申請手続き等が簡素化される方向にある。

さらに、個人の使用のために携行品として一時的に持ち出され、かつ持ち帰られる暗号製品については、規制対象外となった。また、パスワードの保護などの認証・デジタル署名のための暗号機能を持つ装置については輸出許可が不要となった。さらに、公開鍵暗号または 64bit 以下の共通鍵暗号を含む暗号応用の市販のソフトウェアとハードウェアについては、エンドユーザが暗号機能を簡単に変更できないようにしてあるなどの一定の条件を満たしていれば、規制対象外となった。

しかし、市販のソフトウェアについては、従来は暗号の強度によらず規制対象から除外されていたが、64bit を超える共通鍵暗号は新たに規制対象になり、規制が強化された形となっている。また、規制には暗号製品に対応する「特定貨物の輸出規制」と、ソフトウェアが含まれる「特定技術の提供規制」の 2 種類があり、特に「特定技術の提供規制」に注意が必要である。すなわち、特定技術の役務取引許可は特定技術が非居住者に移転することを規制しており、国内に滞在している者でも外為法上居住者と認められていない者が存在しているため、国内における技術の移転であっても役務取引許可が必要な場合が存在す

る。

7.5.2 日本における今後の課題

米国の暗号輸出規制の動向にからみ、日本においてもさらなる規制緩和の議論がある。なお、不正アクセス防止法の成立や住民票基本台帳法改正に伴う、個人情報保護法や消費者保護法の議論、さらに電子署名法の検討など、暗号関連の法令が成立したり、具体的な検討が行われたりしている。

8 付録 - 1　参考文献

2.暗号技術の利用形態

[1] NIST：FIPS PUB 46-3 DATA ENCRYPTION STANDARD (DES) 1999 October 25　URL: http://csrc.nist.gov/fips/fip46-3.pdf

[2] ECOM: 本人認証技術検討WG報告書　評価基準（第１版）、1998

[3] J. Myers、M. Rose：”Post Office Protocol - Version 3”, RFC 1939, 1996, URL: http://www.ietf.org/rfc/rfc1939.txthttp://ds.internic.net/rfc/rfc2040.txt

[4] W. Simpson：” The Point-to-Point Protocol (PPP)”, RFC1661, 1994, URL: http://www.ietf.org/rfc/rfc1661.txt

[5] W.Simpson：”PPP Challenge Handshake Authentication Protocol (CHAP)”, RFC1994, 1996　URL: http://www.ietf.org/rfc/rfc1994.txt

[6] L.Lamport:”Password Authentication with Insecure Communication”, Communications of the ACM, Vol. 24(11), pp. 770-772, November 1981

[7] Krawczyk, H., Bellare, M., and R. Canetti："HMAC: Keyed- Hashing for Message Authentication”, RFC2104, 1997
URL: http://www.ietf.org/rfc/rfc2104.txt

[8] S. Dusse, P. Hoffman, B. Ramsdell, L. Repka：” S/MIME Version 2 Message Specification”, RFC2311, 1998　URL: http://www.ietf.org/rfc/rfc2311.txt

[9] RSA Security 社 PKCS 関連ホームページ：
URL: http://www.rsasecurity.com/rsalabs/pkcs/

[10] N. Freed, N. Borenstein：”Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”, RFC2045, 1996
URL: http://www.ietf.org/rfc/rfc2045.txt

[11] J. Callas, L. Donnerhacke, H. Finney, R. Thayer:”OpenPGP Message Format”, RFC2440, 1998　URL: http://www.ietf.org/rfc/rfc2440.txt

[12] URL: http://www.cert.org/

[13] URL: http://home.netscape.com/eng/ssl3/draft302.txt

[14] T. Dierks, C. Allen：” The TLS Protocol Version 1.0”, RFC2246, 1999
URL:http://ds.internic.net/internet-drafts/draft-rivest-rc2desc-00.txt
http://www.ietf.org/rfc/rfc2246.txt

[15] S. Kent, R. Atkinson：”Security Architecture for the Internet Protocol”, RFC 2401, 1998　URL: http://www.ietf.org/rfc/rfc2401.txt

[16] D. Harkins, D. Carrel:”The Internet Key Exchange (IKE)”, RFC2409, 1998
URL: http://www.ietf.org/rfc/rfc2409.txt

[17] M. Blaze: "A Cryptographic File System for Unix." Proceedings of the First ACM Conference on Computer and Communications Security, Fairfax, VA, November 1993. URL: http://www.crypto.com/papers/cfs.pdf

[18] P Kocher, J Jaffe, B Jun:”Differential Power Analysis”, Proceedings of Crypto ‘99, 1999　URL: http://www.cryptography.com/dpa/Dpa.pdf

3.暗号の仕組みと要素技術

3.1 共通鍵暗号

[1] J. Kelsey, B. Schneier, and D. Wagner, "Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA," ICICS '97 Proceedings, Springer-Verlag, November 1997, pp. 233-246.
URL : http://www.counterpane.com/related-key_cryptanalysis.html

[2] J. Kelsey, B. Schneier, and D. Wagner, "Key-Schedule Crypta-nalysis of 3-WAY, IDEA, G-DES, RC4, SAFER, and Triple-DES," Advances in Cryptology--CRYPTO '96 Proceedings, Springer-Verlag, August 1996, pp. 237-251. URL : http://www.counterpane.com/key_schedule.html

[3] 池野信一・小山謙二：「現代暗号理論」第３章 3.4 電子情報通信学会、1986

[4] NIST：FIPS PUB 46-3 DATA ENCRYPTION STANDARD (DES)1999 October 25 URL : http://csrc.nist.gov/fips/fips46-3.pdf

[5] NIST：FIPS PUB 74 - GUIDELINES FOR IMPLEMENTING AND USING THE NBS DATA ENCRYPTION STANDARD
URL : http://www.itl.nist.gov/div897/pubs/fip74.htm

[6] NIST：FIPS PUB 81 DES MODES OF OPERATION
URL : http://www.itl.nist.gov/fipspubs/fip81.htm

[7] Ascom Systec 社ホームページ：
URL : http://www.ascom.ch/infosec/idea.html

[8] 月刊インタフェース 1997 年９月号、ＣＱ出版社

[9] エレクトロニクス 1996 年５月号、オーム社

[10] 日経バイト 1997 年５月号、日経ＢＰ社

[11] bit1997 年 8 月号、共立出版

[12] 三菱電機(株) MISTY のページ
URL : http://www.mitsubishi.com/ghp_japan/misty/200misty.htm

[13] (株)ローレルインテリジェントシステムズ ホームページ
URL: http://www.lis-fss.co.jp/

[14] R.Rivest：”A Description of RC2 Encryption Algorithm”, RFC 2268
URL: http://www.ietf.org/rfc/rfc2268.txt

[15] R.Rivest：”The RC５, RC５-CBC, RC５-CBC-Pad, and RC５-CTS Algorithms”, RFC 2040,
URL: http://www.ietf.org/rfc/rfc2040.txt

[16] A.Biryukov,E.Kushilevitz: “Improved Cryptanalysis of RC5”, proceedings of Advances in Cryptology – Eurocrypto’98 – LNCS 1403, pp.85-99, Springer Verlag, 1998

[17] 竹内清史,早川珠理,下山武司,辻井重男:共通鍵ブロック暗号 RC5,RC6 に対する correlation attack, SCIS2000-A02

[18] J.L.Massey,"SAFER K-64:A Byte-Oriented Block Ciphering Algorithm", pp. 1-17 in Fast Software Encryption (Ed. R. Anderson), Proceedings of the Cambridge Security Workshop, Cambridge, U.K.,Dec.9-11, 1993, LNCS No.

809, Springer-Verlag (1994)
[19] B.Schneier：URL: http://www.counterpane.com/blowfish.html
[20] E.F.Brickell, D.E.Denning, S.T.Kent, D.P.Maher, W.Tuchman：
"SKIPJACK Review Interim Report", CPSR, 1993
URL: http://www.cpsr.org/dox/program/clipper/skipjack-interim-review.html
[21] "SKIPJACK and KEA Algorithm Specifications", Version 2.0, 29 May, 1998,
URL:http://csrc.nist.gov/encryption/skipjack-kea.htm skipjack.pdf
[22] URL : ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/khufu.tar.gz
[23] URL: http://www.entrust.com/library.htm
[24] URL: http://www.ietf.org/rfc/rfc2144.txt
[25] URL: http://www.jetico.sci.fi/gost.htm
[26] B.Schneier, "Applied Cryptography : Second Edition", John Wiley & Sons, 1996
[27] 角尾幸保,久保博靖,宮内宏,中村勝洋：統計的手法により安全性が評価された暗号, SCIS'98-4.2.B
[28] 角尾幸保,太田良二,宮内宏,中村勝洋：統計的手法に基づく暗号強度評価支援システム, SCIS'98-4.2.A
[29] M8: ALGORITHM REGITER ENTRY | {iso standard 9979 m8 (29)}国内窓口は情報処理振興事業協会(IPA) (crypto@ipa.go.jp)
[30] 時田俊雄: ISO 登録暗号アルゴリズム M8 の強度解析, ISEC99-24(1999-07)
[31] 金子敏信: M8 暗号の弱鍵に関する一考察, ISEC99-25(1999-07)
[32] 佐野文彦,松本勉: M8 暗号に対する一攻撃手法,ISEC99-42(1999-9)
[33] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr., Luke O'Connor, Mohammad Peyravian, David Safford Nevenko Zunic : "MARS - a candidate cipher for AES", July, 17 1998
URL : http://www.research.ibm.com/security/mars.html
[34] Ronald L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin : "The RC6 Block Cipher", Version 1.1 - August 20, 1998
URL : http://www.rsa.com/rsalabs/aes/
[35] Gaël Hachez, François Koeune, Jean-Jacques Quisquater, "cAESar results: Implementation of For AES Candidates on Two Smart Cards", February 4, 1999 URL: http://www.dice.ucl.ac.be/crypto/CAESAR/caesar.html
http://www.dice.ucl.ac.be/crypto/CAESAR/smartcards.ps
[36] Lars R. Knudsen, Willi Meier: "Correlations in RC6", July 29, 1999
URL :http://csrc.nist.gov/encryption/aes/round2/comments/19990812-lknudsen.pdf http://www.ii.uib.no/~larsr/myaes.html/rc6.ps
[37] Joan Daemen, Vincent Rijmen:" AES Proposal: Rijndael", Document version 2, Sept.3, 1999 URL : http://www.esat.kuleuven.ac.be/~rijmen/rijndael/
[38] 小池正彦,佐野文彦,川村信一,斯波万恵:IC カード向け暗号ライブラリの評価,

SCIS2000-D33

[39] Ross Anderson, Eli Biham, Lars Knudsen : “Serpent: A Proposal for the Advanced Encryption Standard”
URL : http://www.cl.cam.ac.uk/~rja14/serpent.html

[40] Eli Biham, “A Fast New DES Implimentation in Software”, Fast Software Encryption 4, 1997
URL : http://www.cs.technion.ac.il/~biham/publications.html CS0891.ps.gz

[41] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson : “Twofish: A 128-Bit Block Cipher”, 15 June 1998
URL : http://www.counterpane.com/twofish-paper.html

[42] 谷口文一,太田和夫,大久保美也子:Triple DES を巡る最近の標準化動向について,日本銀行金融研究所/金融研究/1999.9
URL : http://www.imes.boj.or.jp/japanese/zenbun99/kk18-b1-2.pdf

[43] ANSI X9.52-1998, “Triple Data Encryption Algorithm Modes Of Operation”

[44] NIST AES ホームページ：
URL: http://csrc.nist.gov/encryption/aes/

[45] 宇根正志: 最近の AES を巡る動向について，日本銀行金融研究所 Discussion Paper No.98-J-21
URL http://www.imes.boj.or.jp/japanese/jdps98_index.html

[46] ISO/IEC 10116 “Information technology --- Security techniques --- Modes of operation for an n-bit block cipher”, 2nd edition, 1997

[47] ＲＣ４, URL: ftp://sable.ox.ac.uk/pub/crypto/misc/rc4.tar.gz

[48] C. Adams, J. Gilchrist : “The CAST-256 Encryption Algorithm”, June 1999
URL http://www.ietf.org/rfc/rfc2612.txt

3.2 公開鍵暗号

[1] 岡本龍明，山本博資：「現代暗号」, 1997 年，産業図書

[2] 岡本龍明，太田和夫：「暗号・ゼロ知識証明・数論」, 1995 年，共立出版

[3] D. R. Stinson 著，櫻井幸一監訳：「暗号理論の基礎」、1996 年、共立出版

[4] Bruce Schneier : “Applied Cryptography (Second Edition) ”, 1996, John Wiely & Sons, Inc.

[5] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone : “Handbook of Applied Cryptography ”, 1997, CRC Press
URL: http://www.cacr.math.uwaterloo.ca/hac/

[6] Ian Blake, Gadiel Seroussi, Nigel Smart : “Elliptic Curves in Cryptography ”, 1999, Cambridge University Press.

[7] RSA Laboratories Cryptography FAQ
URL: http://www.rsasecurity.com/rsalabs/faq/

[8] 岡本龍明，内山成憲：「楕円曲線暗号の安全性について」,情報処理 39 巻 12 号, 1998 年，情報処理学会

[9] 岡本龍明，内山成憲：「安全性が証明された新しい公開鍵暗号」,情報処理 40 巻 2 号, 1999 年，情報処理学会

[10] Dan Boneh : “Twenty Years of Attacks on the RSA Cryptosystem ”,Notices of the AMS, Feb. 1999, American Mathematical Society
URL: http://www.ams.org/notices/199902/boneh.pdf
[11] ISO/IEC JTC 1/SC 27 “Information technology - Security techniques ”
URL: http://www.din.de/ni/sc27/
[12] IEEE P1363 Draft: Standard Specifications For Public Key Cryptography
URL: http://grouper.ieee.org/groups/1363/
[13] ANSI X9 Draft
URL: http://grouper.ieee.org/groups/1363/contrib.html#other

3.3 ハッシュ関数

[1] 岡本龍明、山本博資：「現代暗号」1997 年、産業図書
[2] Bruce Schneier : “Applied Cryptography (Second Edition) ”, 1996, John Wiely & Sons, Inc.
[3] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone :
“Handbook of Applied Cryptography ”, 1997, CRC Press
URL: http://www.cacr.math.uwaterloo.ca/hac/
[4] 櫻井幸一監訳：「暗号理論の基礎」、1996 年、共立出版
[5] “Open Design”No.14, 1996 年、CQ 出版
[6] 池野信一、小山謙二：「現代暗号理論」、1986 年、電子情報通信学会
[7] 辻井重男：「暗号」、1996 年、講談社選書メチエ
[8] 電子情報通信学会：「『暗号アルゴリズムの設計と評価』ワークショップ講演論文集」、1996 年、電子情報通信学会
[9] RSA Laboratories Cryptography FAQ
URL: http://www.rsasecurity.com/rsalabs/faq/

3.4 乱数

[1] 宮武修、脇本和昌：「乱数とモンテカルロ法」、1978 年、森北出版
[2] Knuth、渋谷政昭訳：「準数値算法／乱数」、1981 年、サイエンス社
[3] 岡本栄司：「暗号理論入門」、1993 年、共立出版
[4] ザルツブルグ大学乱数研究チームのホームページ：
URL: http://random.mat.sbg.ac.at/
[5] Mersenne Twister 法ホームページ：
URL: http://www.math.keio.ac.jp/~matumoto/mt.html
[6] Intel(R) Random Number Generator ホームページ
URL: http://developer.intel.com/design/security/rng/rng.htm
[7] 東芝ランダムマスターホームページ
URL: http://www.toshiba.co.jp/product/abwr/random/

3.5 電子透かし

[1] 「電子透かし」がマルチメディア時代を守る：日経エレクトロニクス 1997.2.24(no.683)
[2] 松井甲子雄：「電子透かしの基礎－マルチメディアのニュープロテクト技術－」、森北出版(1998)

[3] 「電子情報の不正コピー防止」、日経ビジネス 1998.2.23 pp.68-70.
[4] http://www.dvcc.com/dhsg/
[5] http://www.cidf.org/
[6] 日本電子工業振興協会：「電子透かし技術に関する調査報告書」、1999.3
[7] .A.P. Petitcolas, R.J.Anderson, M.Kuhn："Attacks on copyright marking sytem", 2nd International Information Hiding Workshop, Portland, USA (1998)

4.暗号鍵の管理方式

4.1 鍵のライフサイクル

[1] CCITT,"The Directry-Authentication Framework",X.509,1997

4.2 鍵の配送

[1] 今井秀樹：「暗号のおはなし」、1993 年、財団法人日本規格協会
[2] B.Schneier,“Applide Cryptography 2nd edition”, John Wiley& Sons,
[3] Haudbook05 Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, “Applied Cryptography”, CRC Press

4.5 公開鍵基盤

[1] CCITU-T: Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, ITU-T Recommendation X.509 (1997 E), August 1997.
[2] R. Housley, W. Ford, W. Polk, D. Solo: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC2459, 1999.
[3] C. Adams, S. Farrell: Internet X.509 Public Key Infrastructure Certificate Management Protocols, RFC2510, 1999.
[4] S. Boeyen, T. Howes, P. Richard: Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2, RFC2559, 1999.
[5] M.Myers, R.Ankney, A.Malpani, S.Galperin, C.Adams; X.509 Internet Public key Infrastructure Online Certificate status Protocol – OCSP, RFC2560, 1999
[6] M. Myers, C. Adams, D. Solo、D. Kemp: Internet X.509 Certificate Request Message Format, RFC2511, 1999.
[7] B. Kaliski: PKCS 10: Certification Request Syntax Version 1.5, RFC2314, 1999.
[8] W. Yeong, T. Howes, S.Kille: Lightweight Directory Access Protocol, RFC1777, 1995.
[9] M. Wahl, T. Howes, S.Kille: Lightweight Directory Access Protocol(v3), RFC2251, 1997.
[10] S.Boeyen, T. Howes, P. Richard: Internet X.509 Public Key Infrastructure LDAPv2 Schema, RFC2587, 1999.
[11] R. Housley, P. Hoffman: Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP, RFC2585, 1999.

5.暗号の評価と安全性

[1] A. K. Lenstra and E. R. Verheul、 "Selecting Cryptographic Key Sizes "http://www.cryptosavvy.com/cryptosizes.pdf

[2] 岡本龍明、山本博資：「現代暗号」、産業図書、1997

[3] 池野信一、小山謙二：「現代暗号理論」、電子情報通信学会

[4] 時田俊雄、松井充、反町亨、“暗号解読・強度評価技術、”三菱電機技報 Vol.72 No.5、 pp.8-11、 1998 年 5 月

[5] RSA 社ホームページ： http://www.rsasecurity.com/rsalabs/challenges/

[6] Electronic Frontier Foundation, "Cracking DES," O'Reilly, (1998) http://www.genpaku.org/crackdes/cracking-desj.html(山形浩生氏による邦訳)

[7] Project Bovine ホームページ：http://rc5.distributed.net/

[8] M.Blaze, W.diffie, R.L.Rivest, B.Schneier, T.Shimomura, E.Thompson, M.Wiener,“Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security,” BSA, 1996, http://www.bsa.org/policy/encryption/cryptographers_c.html

[9] B. Schneier, "Applied Cryptography 2nd Ed.," John Wieley & Sons, Inc.,(1996)

[10] E. Biham and A. Shamir, "Differential Cryptoanalysis of DES-like Cryptosystem," Proc. of CRYPTO'90, LNCS537, Springer-Verlag, pp.2-21 (1991)

[11] M.Matsui, "Linear Cryptoanalysis Method for DES Cipher," Proc. of Eurocrypt'93, LNCS765, Springer-Verlag, pp.386-397(1994)

[12] 楠田浩二、櫻井幸一、”公開鍵暗号方式の安全性評価に関する現状と課題” Discussion Paper No.97-J-11 、 日 本 銀 行 金 融 研 究 所 、 1997.7 http://www.imes.boj.or.jp/jdps/97-J-11.pdf

[13] 宇根正志、岡本龍明、“安全性証明付きの公開鍵暗号方式を巡る動向について” SCIS99、 pp.881-886、 1999

[14] 宇根正志、“RSA 署名に対する新しい攻撃法の提案について - Coron-Naccashe-Stern の攻撃法-、”Discussion Paper No.99-J-22、日本銀行金融研究所、1999.6 http://www.imes.boj.or.jp/jdps99/99-J-22.pdf

[15] IPA セキュリティセンター・セキュリティ評価・認証ホームページ： http://www.ipa.go.jp/SECURITY/ccj/index-j.html

6.暗号システムの実装・構築方式

6.3 暗号機能の検証

[1] "FIPS PUB 74, GUIDELINES FOR IMPLIMENTATION AND USING THE NBS DATAENCRYPTION STANDARD",NIST URL: http://www.itl.nist.gov/fipspubs/fip74.htm

[2] "Implementation Guidance for FIPS PUB 140-1 and the Cryptogra-phic Module Validation Program",NIST

URL: http://www.itl.nist.gov/fipspubs/fip140-1.htm

6.4 暗号利用とコスト

[1] B. Schneier：”Why Cryptography is Harder than it Looks” ,COUNTERPANE SYSTEMS　URL: http://www.counterpane.com/whycrypto.html

7. 暗号に関する法令・制度

7.5 日本における制度・法令

[1] 「電子商取引環境整備研究会 中間論点整理」、通産省・電子商取引環境整備研究会、平成９年 11 月。URL: http://www.ecom.or.jp/miti/971127/

[2] 「情報セキュリティ調査研究報告書」、財団法人社会安全研究財団・情報セキュリティ調査研究委員会、平成９年４月１日。
URL: http://www.npa.go.jp/seiankis1/title.htm.

[3] 田渕治樹：「ＯＥＣＤ暗号ガイドラインの概要とその影響」、NIKKEI COMPUTER　1997．5．26.

[4] The OECD Guidelines on Cryptography Policy,
URL: http://www.oecd.org/dsti/iccp/crypto_e.html

[5] 「世界暗号政策ツアー」、WIRED JUNE 1997.

[6] The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption,
URL: http://www.crypto.com/key_study/

[7] Cryptography and Liberty 1999: An International Survey of Encryption Policy, published by the Electronic Privacy Information Center (EPIC) in June., URL: http://www2.epic.org/reports/crypto1999.html

9 付録 - 2　索引

1 付録－3　検討メンバーリスト

ＥＣＯＭ

辻　　秀一　　電子商取引実証推進協議会　主席研究員

重松　孝明　　電子商取引実証推進協議会　主席研究員

リーダー・サブリーダー

渡辺　晋一郎	リーダー	(株)セイコーエプソン無線技術実用化センター　主任
稲村　雄	サブリーダー	日本ベリサイン（株）マーケティング部テクノロジー課課長

メンバー

木村　道弘	日本電気（株）　ミドルウェア事業部第三技術　マネージャー
寺尾　太郎	富士ゼロックス(株)　ニュービジネスセンター　i-Service 開発部
西岡　毅	三菱電機(株)　情報技術総合研究所　情報ｾｷｭﾘﾃｨ技術部　主事
半田　富己男	大日本印刷（株）ビジネスフォーム事業部　ＩＣカード事業本部ソフト開発部
山中　喜義	NTT インテリジェントテクノロジー㈱　第一事業部　技術部長

平成 12 年３月発行
発行：電子商取引実証推進協議会
東京都江東区青海２－４５
タイム２４ビル１０階
Tel　03-5531-0061
E-mail　info@ecom.or.jp