# 暗号入門 1 – 暗号通信と認証

荒川靖弘

第 2 版（Rev. 2） 2002 年 3 月 24 日

## 目次

# 1　暗号通信とは

皆さんの中には「暗号通信」と聞くとスパイ映画や軍や戦争を連想するかもしれません。確かに 1970 年代までの暗号通信は非常に高価なコミュニケーション手段で，これを使うのは政府機関や軍，または大企業の一部に限られていました。しかし今や暗号は安価かつ安全なものになってきていて，中小企業や個人でも使えるレベルにまでなってきています。このような状況を踏まえ，本テキストでは暗号通信を以下のように考えます。

> 「暗号通信とは，特定の相手とのコミュニケーションにおいて，安全でない通信経路を使って安全な通信手段を提供する要素技術である」

## 1.1　「安全な通信」のための 4 つの条件

「特定の相手」と「安全な通信」を行うためには，以下に示す 4 つの条件が必要です。

1. 機密性（Confidentiality）
2. 完全性（Integrity）
3. 認証（Authentication）
4. 否認防止（Non-repudiation）

これらの条件についてもう少し詳しく説明しましょう。

### 1.1.1　機密性

通信中は，その内容を第 3 者に「盗聴」されないようにする必要があります。

「盗聴」というと電話盗聴のようなものを連想しがちですが，今日インターネットで可能な「盗聴」はもっと巧妙で効率的でしかも大規模なものです。盗聴されている人は大抵の場合そのことに気がつきません。盗聴の結果は「ログ」として無限にしかも半永久的に記録されます。そしていつでもログから必要な情報を検索・抽出できます。現在の通信データは全てディジタル化されているため，保存も検索も（人間が直接「盗聴」していた時代に比べて）最小限のコストで最大の効果が期待できます。（ログさえあれば 5 年前に貴方が不倫相手に送った電子メールのラブレターを検索するなど簡単なことです）

### 1.1.2　完全性

インターネットを流れるデータをインターセプトすることは可能です。（「やろうと思えばできる」程度の難しさ）このことを利用して通信内容の「改竄」を行うことができます。「改竄」が巧妙であれば通信中にそれと気づくことは難しいでしょう。（気づくのは何がしかの被害・損害を被った後です）

暗号通信では通信内容の完全性を保証する手段が必要になります。これは暗号を応用した「電子署名」（digital signature）を導入する事で解決できます。（3章で詳しく説明します）

### 1.1.3 認証

更に通信をインターセプトし，他者への「なりすまし」を行うこともできます。なりすましを防ぐためには，お互いの相手が「本人」であることを保証（つまり「認証」）するための手だてが必要です。

顔の見えない相手をいかにして「認証」するかというのは難しい問題です。更に認証に必要な情報は第 3 者に対して「機密」でなければなりませんし，認証プロセスは「完全」でなければなりません。

認証プロセスの一部は暗号と電子署名を使って達成できますが，厳密に行うためには暗号以外の手段を併せて導入する必要があります。

### 1.1.4 否認防止

「否認防止」とは通信上のある情報を送信した事を否認できないことを意味します。「完全性」と「認証」を達成する事で「否認防止」も達成できます。相手から送られてきたメッセージに相手の（本物の）署名が添えられていれば，そのメッセージを送ったという事実は否定できません。

この条件は見落とされがちですが，通信上で契約（または単に約束）を交わす場合などには必須の条件となります。

## 1.2 暗号通信を支える技術

以上 4 つの条件について考えましたが，暗号通信の鍵となる技術としては「暗号」，「電子署名」，「認証」があることがわかると思います。（電子署名も暗号の一種ですが本テキストでは分けて考えます）次の章からはそれぞれの技術についてもう少し詳しく解説していくことにします。

# 2 暗号

現在の暗号通信で使われる暗号は，専ら数学的な手法を用います。（コンピュータネットワークに組み込むにはその方が便利です）

$$S' = F(S, K_1) \tag{1}$$
$$S = F^{-1}(S', K_2) \tag{2}$$

元のデータ列 $S$ に対し関数 $F$ をパラメータ $K_1$ と共にあてはめ，新たなデータ列 $S'$ を生成します。（メッセージ暗号の場合は元のデータ列を「平文」，新たなデータ列のことを「暗号文」と呼びます。）新たなデータ列から元のデータ列を推測できないのが特徴です。このとき，関数 $F$ にあたる部分を「暗号方式」または単に「アルゴリズム」（algorithm）と呼び，パラメータにあたる部分（$K_1$ または $K_2$）を「鍵」（key）と呼びます。また，元のデータ列から異なるデータ列に変換する（式 1）ことを「暗号化」（encryption）と呼び，暗号化したデータ列から元のデータ列に復元する（式 2）ことを「復号化」（decryption）と呼びます。

よくできた暗号の暗号強度はアルゴリズムではなく鍵（厳密には鍵のサイズ：鍵長）に依存するように設計されています。アルゴリズムは「コード」としてシステムに組み込まれるため，リバースエンジニアリング等で容易に知られてしまいます。このとき，暗号強度をアルゴリズムに依存した設計にしていると簡単に暗号が破られることになります。

現在広く知られている暗号方式は大きく 2 つに分類されます。以降では 2 つの暗号方式について登場順に解説します。

## 2.1 秘密鍵暗号

秘密鍵暗合はもっとも古典的で現在もよく使われる暗合方式です。古典的なものでは「シーザー暗号」が有名でしょう。これはアルファベットの並びを決められた数だけずらす換字暗号です。例えば「順方向に 3 つ」（これが鍵になります）ずらすと決めれば，「YES」は「BHV」になります。

このように，秘密鍵暗号（secret key cryptography）は暗号化と復号化に同じ鍵を用います。（式 1,2でいうと $K_1 = K_2$ となります）また通常，復号化は暗号化の逆手順になります。暗号化と復号化の手順および鍵が本質的に同じなので「対称暗号」（symmetric cryptography）と呼ばれることもあります。

## 2.2 秘密鍵暗号の暗号方式

秘密鍵暗号で現在広く使われている暗号方式をいくつか紹介し，特長と問題点について考えてみます。

### 2.2.1 DES

「DES」[*1]（Data Encryption Standard）は，1975 年に NBS（National Bureau of Standards and Technology, 後の NIST: National Institute of Standards and Technology: 国家標準技術研究所）で公表され，1977 年以降米国連邦政府の標準として採用されている暗合方式です。もともとは IBM が考案した暗合方式「Lucifer」を元に NSA（National Secrurity Agency: 国家安全保障局）が改良したいわくつきの暗合方式です。DES では 56bit 長の鍵を使いますが，公開当初から 56bit の鍵長は短すぎるといわれていました[*2]。実際に RSA Security 社が行った「DES challenge III」（DES 破りコンテストの 3 回目，1999 年 1 月開催）では僅か 1 日で鍵を特定されています。現在は，DES の脆弱性を補うため，DES を 3 回かける「Triple DES」（この方法により鍵長を 128bit にできます）がよく使われますが，次に説明する AES に標準の座を明け渡しました。

### 2.2.2 AES

1997 年，NIST は時代遅れになりつつある DES に代わる新しい暗合方式「AES」（Advanced Encryption Standart）を一般から募集し，2001 年 11 月に Rijndael が正式に採用されました[*3]。AES では 128,192,256bit の鍵長を選択できます。既に AES は様々な暗号ツールに組み込まれています。

### 2.2.3 RC

「RC」（Rivest Code）は RSA Security 社が提供する暗合方式のシリーズです。最新[*4] の RC6 は一時 AES の候補として上がっていました。RC シリーズは Ronald Rivest （2.4.2章で紹介する「RSA」の発明者のひとり）

[*1] FIPS PUB 46-2: http://www.itl.nist.gov/fipspubs/fip46-2.htm 参照。
[*2] http://www.genpaku.org/crackdes/cracking-desj.html 参照
[*3] FIPS PUB 197: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf 参照
[*4] RC シリーズは RC9 まで商標登録されているそうです。

が RSA Security 社のために設計したものです。RC シリーズの暗合方式は商業分野で（主に R2, R4 ですが）広く使われています。

### 2.2.4 その他の秘密鍵暗号

他にも IDEA, CAST5, Blowfish, Twofish といった暗号方式がよく使われます。

「IDEA」（International Data Encryption Algorithm）はスイスの Ascom 社が提供する暗合方式です。（現在は子会社である iT_SEC iT_Security AG 社が管理しているそうです）IDEA は非商用[*5] に限り自由に使っていいことになっています。

### 2.2.5 秘密鍵暗号の特長と問題点

以上述べた暗号方式は，コンピュータシステムへの実装が容易で処理が速い，という特長があります。一方，秘密鍵暗号には問題も色々あります。そのうち最も根本的な問題は「通信を行う者同士が鍵についてあらかじめ合意しなければならない」というものです。合意の過程で鍵についての情報が第 3 者に漏れれば暗号は意味をなさなくなります。

1970 年代まではこの問題を解決するために主に物理的な手段を使っていました。例えば「鍵情報を鍵のかかった鞄に入れ，それを密使に手錠で繋いで輸送し，相手（の密使）に手渡す」といった具合にです。

更に秘密鍵暗号の運用上の問題点としては，秘密鍵暗号は 1 対 1 通信で使われる暗号であるため，（政府や軍や企業など）組織全体で運用する場合には必要となる鍵の数が膨大になり過ぎる，という点が挙げられます。（$n$ 人が秘密鍵暗号でやり取りするには一番単純な場合で $n(n-1)/2$ 個の鍵が必要になります）

## 2.3 公開鍵暗号

秘密鍵暗号が抱える問題を解決するため，暗号学者達は以下に示す命題について考えるようになりました。

> 「第 3 者が監視しているかも知れない回線において，2 人があらかじめ何らかの合意をすることなく，安全に通信する方法はあるか」

この命題に対する答えが「公開鍵暗号」（public key cryptography）です。公開鍵暗号では「公開鍵」（public key）と「秘密鍵」（secret key）の 2 つの鍵を用います。（2 つの鍵を合わせて「鍵ペア」（key pair）と呼ぶことがあります）2 つの鍵はお互いに関連していますが，公開鍵からは秘密鍵を導き出せないような構造になっています。通信相手に任意の方法で公開鍵を渡し，その公開鍵でデータを暗号化します。暗号化されたデータは秘密鍵を持っている者だけが複合化できます。暗号化と復号化に異なる鍵を使うため「非対称暗号」（asymmetric cryptography）と呼ばれることもあります。

「2 つの鍵はお互いに関連しているのに公開鍵からは秘密鍵を導き出せない」というロジックは一見不自然に見えますが，ある数学的特徴（NP（Nondeterministic Polynomial-time）問題と呼ばれます）を使えば比較的簡単に実現できる事が分かりました。

---

[*5] 「非商用」という言い回しのため誤解している人が多いですが，ふつう「商用」とは全ての営利活動に関る業務一般を指し，「非商用」とは「商用」でない活動を指します。従って会社で使うのであれば，どのような形態であれ「商用」と見なされます。

## 2.4 公開鍵暗号の暗号方式

公開鍵暗号で現在広く使われている暗号方式をいくつか紹介し，特長と問題点について考えてみます。

### 2.4.1 Diffie-Hellman の鍵交換

Whitfield Diffie と Martin Hellman の 2 人が 1976 年に発表した鍵交換のアルゴリズムが，公開鍵暗号の草分けとなりました。この方式は先の命題の直接的な答えになっていて，2 人が暗号通信（この場合の暗号は秘密鍵暗号です）を開始する際に鍵情報そのものを配信することなく 2 人が同じ鍵を取得できます。

Diffie-Hellman の鍵交換は「離散対数問題」が数学的に困難である事に依っています。すなわち各々秘密情報 $X_1, X_2$ を選び，その値を元に $Y_1, Y_2$ を求めます。（$X$ を累乗して素数で割った余りを求めます。累乗する値と素数はあらかじめ合意（公開してもよい）しておく必要があります）更に $Y_1, Y_2$ をお互いに交換し，交換した値と手元の秘密情報を元に共通の値 $K$ を導出します。

$$K = F(X_1, Y_2) = F(X_2, Y_1) \tag{3}$$

「離散対数問題」により $Y_1, Y_2$ から元の値 $X_1, X_2$ を求めることは非常に困難であり，従ってそれを元に求められる $K$ の特定も困難となります。値 $K$ はそのまま秘密鍵暗号の鍵として用いることができます。

Diffie-Hellman の鍵交換のアルゴリズムは非常によくできていると言われていますが，通信を行う 2 者がリアルタイムに協調作業を行う必要があります。リアルタイム通信で暗号化セッションを張るのには便利ですが，電子メールのようなメッセージの暗号には向きません。

Diffie-Hellman の鍵交換に関する特許は 1997 年 4 月 29 日を以って失効しました[*6]。Diffie-Hellman の鍵交換は SSH2（5.6章参照）などで使われています。

### 2.4.2 RSA

Ronald Rivest, Adi Shamir, Len Adleman の 3 人が 1977 年に発表した暗号方式「RSA[*7]」は「2 つの大きな素数の積を計算して合成数を作るのは簡単だが，その合成数を元の素因数に分解するのは非常に難しい」という事実を利用しています。

RSA では 2 つの素数から秘密鍵情報と公開鍵情報を作成します。鍵の生成について通信相手と協調作業する情報がないため（あらかじめ作っておいた公開鍵を配布すればよい），RSA は電子メールなどのメッセージ暗号に向いています。

RSA は PGP（5.1章参照）との特許関係の係争などで一躍有名になりましたが，2000 年 9 月 20 日の特許[*8]の失効をもってパブリックドメイン化されました。

---

[*6] 日本には「Diffie-Hellman の鍵交換」に関する特許はありません。
[*7] この名は Ronald Rivest, Adi Shamir, Len Adleman の 3 人のイニシャルからきています。
[*8] 日本には RSA の特許はありません。

### 2.4.3 ElGamal

Diffie-Hellman の鍵交換のアルゴリズムを応用して別の暗号方式が考えられます。「ElGamal」と呼ばれるこの方式は，RSA と同じくメッセージ暗号に使えます。

ElGamal は Diffie-Hellman の鍵交換に関する特許の失効（1997 年 4 月 29 日）を以ってフリーになりました。RSA よりも早い時期に特許が失効したため，PGP（5.1章参照）等を中心によく使われています。

### 2.4.4 公開鍵暗号の特長と問題点

公開鍵暗号の特長はなんといっても「強力」であることです。外部に暗号解読のヒントとなるような情報を持ち出す必要がないため，クラッキングをより困難なものにできます。また理論的には鍵長をいくらでも長くすることができるので，コンピュータのスペックの向上によるクラッキングの危険にも柔軟に対応できます。

一方，公開鍵暗号方式にも問題がないわけではありません。実装上の問題として，計算集約的なロジックを用いるため鍵の生成や暗号化/復号化の処理に時間がかかる，という点が挙げられます。

そこで公開鍵暗号と秘密鍵暗号の長所を組み合わせる「ハイブリッド暗号」がよく使われます。

## 2.5 ハイブリッド暗号

ハイブリッド暗号では秘密鍵暗号鍵の配布プロセスに公開鍵暗号を用い，実際のコミュニケーションは配布された鍵を使って秘密鍵暗号で高速に行います。この方法を使うことにより毎セッションごとに秘密鍵暗号鍵を変える（この鍵のことを特に「セッション鍵」（session key）と呼びます）ことが可能になります。セッション鍵を用いる事によりある時点の鍵についてクラッキングできる時間が短くなるので，相対的に秘密鍵暗号の暗号強度も増します。

## 2.6 鍵配布の問題

秘密鍵暗号における鍵の管理と配布の問題は 2.2.5章で述べたとおりですが，公開鍵暗号を使っても完全には解決しません。すなわち，公開鍵の配布が簡単な公開鍵暗号では受け取った公開鍵が確かに本人のものであるかどうか確認する方法がないため，公開鍵そのものが「改竄」や「なりすまし」を受けてしまう可能性があります。

もちろん，公開鍵を渡すプロセスに対しても物理的な安全措置を講じれば解決できるかもしれませんが，それでは私達が使えるような「安価」な暗号通信は実現できなくなります。この問題を解決するには，メッセージの発信者を特定できる仕組み（署名と認証）を組み込む必要があります。

# 3 電子署名

暗号メッセージを送る際，そのメッセージに発信者が間違いなく本人のものである事を示す「印」が付いていれば，受信者はそのメッセージが「本人」のものであると判断することができます。この「印」のことを「電子署名」（digital signature）と言います。

電子署名の仕組みには実は公開鍵暗号（「Diffie-Hellman の鍵交換」以外）を応用できます。すなわち，お互

いに既知である情報に対して秘密鍵で暗号化を行います。暗号化情報を受け取った方はその情報を公開鍵で復号化し内容を検証します。この方法では暗号化できるのは秘密鍵を持っている人のみなので，検証内容が正しければその「署名」した人物が特定できます。

「既知の情報」としては，発信するメッセージの「ハッシュ値」を使うのが合理的です。ハッシュ（暗号の世界では「メッセージ要約」（messeage digest）と呼んでいます[*9]）に使われる方式は「一方向ハッシュ関数」（one-way hash function）とも呼ばれていて，非常に高速でハッシュ値の衝突が少ない[*10] のが特徴です。

なお電子署名に関しては，ドイツの Claus P. Schnorr による電子署名規格の特許が関係していると言われています。この特許は世界中で有効になっており期間も 2008 年までと長いものになっています。

## 3.1 電子署名の方式

現在広く使われている電子署名の方式をいくつか紹介します。

### 3.1.1 RSA

RSA のアルゴリズムをそのまま利用して電子署名として応用できます。RSA による電子署名は商業分野を中心に広く使われています。ただし近年では，セキュリティ上の観点から，公開鍵暗号と電子署名で同じアルゴリズムや鍵を使わない傾向になってきています。

### 3.1.2 ElGamal

ElGamal もそのまま電子署名として応用できます。RSA の場合と同じく，近年では電子署名としてはあまり使われません。

### 3.1.3 DSA

DSA（Digital Signature Algo）は NIST が 1994 年に DSS[*11]（Digital Signature Standard）として公表した方式で，現在でも標準（の 1 つ）として使われています。DSA は ElGamal 署名の改良版で，署名データがなるべく短くなるように設計されています。後に DSA の開発に NSA が関係している事が発覚し問題になりました[*12]。

## 3.2 メッセージ要約の方式

更に現在広く使われているメッセージ要約の方式をいくつか紹介します。

---

[*9] もともとのハッシュ（hash）の意味は「細切れに（調理）する」です。

[*10] ここでいう「衝突が少ない」とは，ハッシュ値から元のデータ列を推測するのが困難であり，更に同じハッシュ値を持つ異なるデータ列を見つけるのが困難である，ことを示します。

[*11] FIPS PUB 186-2: http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf 参照。

[*12] 更に 2001 年 2 月にセキュリティ上の欠陥が発覚し，同年に修正されています。

### 3.2.1 MD4, MD5

Ronald Rivest が考案した方式で，現在も広く使われています。MD5 は MD4 の改良版です。ただし，MD5 にはいくつかの脆弱性が指摘[*13] されており，暗号の最先端分野ではあまり使われなくなってきています。

### 3.2.2 SHA

SHA（Secure Hash Algorithm）は SHS[*14]（Secure Hash Standard）として DSA とほぼ同時期に公表された方式です。SHA は MD4 の影響を受けていると言われますが，ハッシュ値が 160bit と長く，MD4, MD5 より優れていると評価されています。現在は修正された SHA-1 が使われています。

ただしこのアルゴリズムの開発にも NSA が関与しており，「裏口」等の懸念が指摘されています。（今のところそのようなものは見つかっていません）

### 3.2.3 その他のメッセージ要約

上記以外の方式としては（暗号学者の間で評価の高い） MD-RIPE/160 などが使われ始めています。

# 4 認証

電子署名を使う事で「改竄」を防ぐことはできますが，「なりすまし」を防ぐことはできません。（暗号通信を行おうとする Alice と Bob の間に Chris がインターセプトする場合を考えてください。Chris は 2 種類の鍵ペアを用意する事でお互いの相手になりすますことができます）また，電子署名は公開鍵暗号を応用しているため暗号化/復号化のコスト（主に計算量）が高く，リアルタイム通信を行う場合は向きません。

厳密に相手を認証するなら現在知られている暗号方式だけでは無理があります。そのため暗号以外の方法も使わざるをえません。以下にいくつか紹介します。実際にはこれらの方法の「どれか」を使うのではなく，複数の方法を組み合わせて認証プロセスの強化を図ります。

## 4.1 サーバ-クライアント環境での認証

まずは，古典的なサーバ-クライアント環境での認証についていくつか挙げます。サーバ-クライアント環境は閉じたネットワークでは今でもよく使われる方法です。認証に必要な情報をサーバ側で（しかもサービスごとに別々に）管理するため，規模が大きくなるほどセキュリティの維持・管理が難しくなります。

### 4.1.1 パスワード認証

ホストにあらかじめアカウント情報（ユーザ ID とパスワード）を登録しておき，暗号化セッション上で「パスワード」を使って認証を行います。この方法で一旦認証されればセッションが終了するまでユーザの身元を

[*13] ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf 参照。
[*14] FIPS PUBS 180-1: http://www.nist.gov/itl/div897/pubs/fip180-1.htm 参照。

保証できます。

パスワード認証は導入が簡単なのでよく使われますが，認証情報として見た場合の「パスワード」はシンプル過ぎるため，複製が簡単で漏洩しやすく信頼性が高いとはいえません。登録・確認・変更・破棄といった管理が繁雑になりやすいのも問題です。

#### 4.1.2 バイオメトリクス認証

パスワードの代わりに指紋や瞳の虹彩（iris）といった身体的な特徴を認証情報として用います。パスワード認証に比べて認証情報を複製しにくいため「なりすまし」の危険は減ります。しかし認証に身体情報を使うため，（マトリクスのコピーを取られる[*15] などの）トラブルが発生しても認証情報を変更できない，という問題があります。またこの認証システムには「誤差」が存在し，かつ精度はアーキテクチャに依存するため，暗号通信の観点からはあまりいい方法とはいえません。

#### 4.1.3 チャレンジ・レスポンス認証

パスワード認証の発展系ですが，あらかじめ決められたパスワードを使うのでなく，公開鍵と「チャレンジ」（challenge）と呼ばれる任意のデータ列を用います。なお公開鍵はアカウント情報としてあらかじめ登録されていることが必要です。（身元さえ確実ならば公開鍵のコピーをとられても構いません）

暗号化された通信経路が確立した後，ユーザはまずユーザ ID をホストに知らせます。ホストは擬似乱数から「チャレンジ」を生成しユーザ ID に対応する公開鍵で暗号化します。もし相手ユーザが正しければ「チャレンジ」を復号化し正しい結果を返せる筈です。

この方法を使えばパスワード情報をホストに登録する必要がありませんし（つまり認証情報管理が簡単になり漏洩の危険性が減ります），通信経路上に認証情報を流すことなく認証手続きを行うことができます。

#### 4.1.4 ワンタイムパスワード認証

これもパスワード認証の発展系です。ユーザは 1 回限りで使い捨てのワンタイムパスワード（one-time password）を使って認証を行います。ワンタイムパスワードは数学的な方法によって導出します。したがって，ホストとユーザで導出アルゴリズムやパラメータ（トークン）を合わせるための仕組みが必要です。ユーザレベルでの扱いが簡単な割りに強力なので大企業などを中心に導入されています。

### 4.2 ピア・トゥ・ピア環境での認証

特定のホストを持たないネットワークシステムを「ピア・トゥ・ピア」（peer-to-peer）と言います。もともとは（サーバすら不要な）ごく小規模なネットワークでのみ使われていましたが，インターネットのようなホスト分散型のネットワーク環境でも有効です。ピア・トゥ・ピア環境で最も身近なものとしては電子メールが挙げられるでしょう。また電子商取引でも（企業側のコスト負担が少ない）ピア・トゥ・ピア環境が注目されはじめてきていて，この環境で有効な認証システムの整備が進んでいます。

[*15] ちなみに指紋は比較的簡単に偽造できることが知られています。

#### 4.2.1 直接会見による認証

公開鍵を渡すプロセスを通信経路上ではなく直接会って手渡しすれば確実です。仮に公開鍵のコピーをとられたとしても，ハイブリッド暗号と電子署名を組み合わせれば「改竄」も「なりすまし」も防げます。「顔見知り」の小規模なコミュニティであれば殆どの場合この方法で解決できます。ただし，不特定多数の相手とコミュニケーションを図る（商取引などの）場合には効率が悪くなります。

#### 4.2.2 第 3 者機関による公開鍵の配布

通信するもの同士が直接会うことができない場合，信頼できる第 3 者を介して公開鍵を交換します。第 3 者機関の何を以って「信頼できる」とするのかは難しいところです。これについては 5章で具体例を挙げて説明します。

# 5 実装例

さて，いよいよ暗号通信および認証システムの具体的な実装例を紹介していきましょう。

## 5.1 PGP/OpenPGP

もともと PGP（Pretty Good Privacy）は Phil Ziemmermann により，当時米国内で高まりつつあった反核運動の一環として，あるいは米国政府の暗号通信政策に対する防衛手段として開発・公開されました。もちろんそれまでにも企業を中心として暗号ツールを組み込んだシステムは存在していましたが，PGP は「個人レベルで使える強力でフリーな暗号ツール」という点で画期的でした。一方で PGP は（Ziemmermann が当時特許に関して迂闊だったこともあり）重大な特許問題を含んでいたため長い間企業からは敬遠されていました[*16]。PGP は現在 Network Assciates 社 がサポートを行っていますが，2001 年 10 月から事業部門の再構築により開発・販売を中止しています。したがって，PGP を商用で利用するのは難しい情勢です。

一方 PGP の設計思想を継承しつつ，より一般的な仕様としてまとめられたのが OpenPGP[*17] です。OpenPGP に基づきライセンスの問題を排除した実装としては GnuPG（GNU Privacy Gaurd）があります。GnuPG は GPL（GNU General Public License）に基づいて配布されており，商用でも利用できます。

PGP そのものはハイブリッド暗号を使ったメッセージやファイルの暗号化/復号化ツール[*18] ですが，ユーザの認証モデルに際だった特徴があります。PGP の認証モデルは「信用の輪」（web of trust）と呼ばれています。「信用の輪」は 4.2.1章で説明した「顔見知り」同士の信頼関係を基本にしています。「顔見知り」同士がお互いの公開鍵に「署名」する事により相手の公開鍵が有効である事を「保証」します。

例として Alice と Bob がお互いの公開鍵に署名しているとします。ここで Bob とは知り合いですが Alice のことは知らない Chris が Alice の公開鍵を取得します。Chris は Alice のことは知らないのでその鍵が本物かどうか分かりませんが，「顔見知り」である Bob の署名があるため Alice の鍵も有効であると判断できます。

---

[*16] この辺のいきさつについては文献 [3] やインタビュー記事（http://www.zdnet.co.jp/news/0201/28/e_pgp.html）が参考になります。

[*17] RFC 2440: http://www.gnupg.org/rfc2440-ja.html 参照。

[*18] 拙著 http://hp.vector.co.jp/authors/VA023900/note/pgp_tips2.html でも簡単に説明しています。

この例では Bob を通じてお互いに顔見知りでない Alice と Chris が「信用の輪」を結ぶことができます。

PGP ではインターネット上に複数の鍵サーバを置いて互いに同期させています。ユーザは最寄りの鍵サーバから世界中のユーザの公開鍵を取得することができ，公開鍵に添付してある「署名」によってその鍵の有効性を判断します[*19]。

「信用の輪」は非常にシンプルで直感的な方法ですが，大規模なシステム（電子商取引など）では運用が難しい面があります。

## 5.2 X.509

もう一度「信用の輪」で登場した Alice, Bob, Chris の例を考えてみます。この例では見ず知らずの Alice と Chris が「Bob と顔見知りである」という関係をもってお互いに信頼関係を結ぶことができました。ここでもし Bob が人（個人）ではなく公の組織だとしたら，もっと大規模な認証システムを構築できるはずです。

このような発想から ITU-T（International Telecommunication Union, Telecommunication Standardization Sector）で提案されたのが X.509（ISO 9594-8）です。更に IETF（Internet Engineering Task Force）でも同等の規格[*20] が公表されています。もともと X.509 は X.500 ディレクトリシステムを念頭に作られたものでしたが，より柔軟な運用ができるよう改良が加えられています。（現在はバージョン 3）

X.509 の特徴は CA（Certificate Authority: 認証局）と呼ばれる発行機関を置き，CA が署名の正当性を保証する点です。更に複数の CA がツリー型の階層構造になっているため，大規模なシステムに対応できます。しかし運用コストが高いため，小規模のシステムでは扱いにくい面があります。

## 5.3 PKI

インターネットを中心に公開鍵暗号による認証システムの需要が高まり，そのインフラとしての PKI（Public Key Infrastructure: 公開鍵基盤）の整備が急がれています。日本でもいわゆる「電子署名法[*21]」（2001 年 4 月施行）を受けて GPKI（Government Public Key Infrastructure: 政府認証基盤）が発足[*22] しました。PKI は暗号方式，電子署名（「電子証明書」と呼ばれます），ディレクトリ，鍵管理，タイムスタンプ，といった広範なサービスを含んでいます。認証モデルとしては，先に説明した X.509（5.2章）や OpenPGP（5.1章）が有力視されています。特に X.509 は電子商取引と相性が良いため，商業分野で非常に注目されています。

PKI がうまく運営できるかどうかはプロバイダ・ユーザ双方の運用ポリシーとその履行にかかっています。民間レベルでは VeriSign 社[*23] などがかなり運用実績を積んでいます。

## 5.4 Kerberos

Kerberos は MIT（Massachusetts Institute of Technology: マサチューセッツ工科大学）のアテナプロジェクト（Project Athena）で開発されているオープンネットワークシステムのための認証システムです。現在では様々なプラットフォームに実装されています。

乱暴な言い方をすれば Kerberos は PKI のイントラネット版ということができます。企業内でのイントラ

---

[*19] もうひとつ公開鍵の有効性を判断する材料として鍵指紋（key fingerprint）があります。これは公開鍵自体のハッシュ値です。

[*20] RFC 2459: http://www.ipa.go.jp/security/rfc/RFC2459JA.html 参照。

[*21] http://www.meti.go.jp/policy/netsecurity/digitalsign.htm 参照。

[*22] http://www.gpki.go.jp/ 参照。

[*23] http://www.verisign.com/ または http://www.verisign.co.jp/ 参照。

ネット整備が進むにつれ，サーバの分散化とその運用コストが馬鹿にならないものとなってきました。認証に関しては，従来のサーバ-クライアントの環境での認証方式（4.1章参照）が（管理の繁雑さゆえ）殆ど破綻しかけていて，もっと安全で効率的な認証方式が求められています。Kerberos 認証はその鍵となる技術と言えます。

## 5.5 SSL/TLS

SSL（Secure Sockets Layer）はもともと Netscape 社が自社の WWW サーバおよびブラウザに採用した暗号化ソケットのプロトコルです。現在はそのバージョン 3[*24] が公表され，更にこれをもとにした TLS（Transport Layer Security）が IETF で規格化[*25] されています。SSL/TLS は（WWW などの）サーバ・クライアント環境のアプリケーションとの相性がよく，商業レベルの Web アプリケーションでもよく使われます。

オープンソースの分野では OpenSSL の実装が進んでいます。OpenSSL[*26] では SSL v2/v3 と TLS v1 を含んでいます。

SSL/TLS では X.509（5.2章）ベースのサーバ認証とクライアント認証（オプション）をサポートしています。しかし実際にクライアント認証を行っているアプリケーションはほとんど無いようです。この場合，他の認証システムが必要になってきますが，巷でよく見る Web アプリケーションでは SSL で暗号化セッションを張った後にパスワード認証（4.1.1章）でクライアントを認証しています。

## 5.6 SSH/OpenSSH

SSH（Secure SHell）は UNIX 系プラットフォームで使われているリモートアクセス用のシェルツール（いわゆる r 系シェルツール）の暗号化バージョンで，最近は他のプラットフォームでも実装されはじめています。

初期の SSH は Tatu Ylönen が開発し，フリーで公開されていました。その後 Ylönen が開発した SSH にはライセンス上の制限が設けられ，現在は SSH Communications Security 社で開発・販売が続けられています。一方，かつてのフリー版の SSH を元にしたフリーソフトも（SSH とは全く独立に）登場し，最終的には OpenSSH[*27] という形で開発が続けられています。SSH はライセンスが複雑になり過ぎ，企業側でも敬遠されつつあります。一方 SSH Communications Security 社は「SSH」が商標であると主張していて，OpenSSH 開発側に「SSH」の名称を使うのを止めるよう警告しています。

SSH/OpenSSH では公開鍵暗号/電子署名を使ったサーバ-クライアント環境の相互認証（4.1章参照）をサポートしていますが，SSL/TLS が X.509（5.2章）ベースなのに対し，SSH/OpenSSH では PGP（5.1章）の鍵を使うこともできます。更にホストベース認証（サーバ間で人手を介さず相互認証できる仕組み）もサポートしているため，ホスト分散型の環境にも一部（サーバ間の同期など）対応できます。

SSH/OpenSSH の別の使い方としては，TCP ポート転送機能を利用して，SSH サーバをゲートウェイとした VPN（Virtual Private Network）の構築が考えられます。

---

[*24] http://www.netscape.com/eng/ssl3/ 参照。
[*25] RFC 2246: ftp://ftp.isi.edu/in-notes/rfc2246.txt 参照。
[*26] http://www.openssl.org/ または http://www.infoscience.co.jp/technical/openssl/ 参照。
[*27] http://www.openssh.com/ 参照。

## 5.7 実装の失敗例

暗号化通信はいくつかの要素技術の組み合わせで成り立っており，それらの組み合わせや運用を誤れば，暗号化していない状態よりも更に悪い結果をもたらす場合があります。（ユーザは暗号化されているという安心感があり，暗号化通信環境の中ではより無防備になりがちだからです）ここでは，そういった「失敗例」を紹介します。

### 5.7.1 CSS

1999 年末，ノルウェーの何人かの Linux ハッカーが「DeCSS」と呼ばれる DVD デコーダを作り，DVD メディアで使用されている暗号機構（「CSS」（Content Scrambling System）と呼ばれる非公開の方式です）をクラック[*28] してしまいました。

何故 CSS は破られてしまったのでしょう。DeCSS を作ったハッカーが凄腕だったのでしょうか。そうではありません。実際にはハッカーが凄腕だったからではなく（確かに暗号に疎い私達からみれば凄腕かもしれませんが），CSS が「脆弱」だったことが真の原因です。

もともと CSS は著作権保護の目的で実装されました。しかしコンテンツを（厳密にはコンテンツを復号化する鍵を）復号化するための「解除鍵」は DVD プレイヤー内にハードコーディングされているためクラッキングはそれほど難しくありません。しかも単純にコンテンツを不正コピーするだけなら DVD をビット単位で丸ごとコピーすればいいだけなので（DVD プレイヤーには本物か海賊版かを見分ける方法がありません），著作権保護としては CSS は全く機能していないことになります。更に復号化鍵自体が 40bit と非常に短かったのと鍵生成のアルゴリズムや DVD プレイヤー側の鍵管理の方法がずさんだったため，ひとつの鍵が見つかれば芋づる式に他の鍵も推測できてしまうというお粗末さでした。

DVDCCA（DVD Copy Control Association）はこの件に関し「技術者側を取り締まる」ことで問題をかわそうとしています。（5.7.3参照）

### 5.7.2 WEP

現在広く使われている無線 LAN（IEEE802.11b）には「WEP」（Wired Equivalent Privacy）と呼ばれる簡単な暗号機構が備わっています。2001 年冬，ISAAC（Internet Security, Applications, Authentication and Cryptography）は WEP の脆弱性に関する論文[*29] を公表しました。それまでも WEP の脆弱性は指摘されていましたが，この論文により決定的になってしまいました。

IEEE802.11b を策定したワーキンググループは WEP の暗号強度についてあまり重要視してなかったようです。しかし，製品段階で「有線並みのセキュリティがある」などと宣伝したため（WEP という名前がまさにそれを指します），ユーザに対して過度の期待感を与えることになってしまいました。

WEP の問題点は暗号鍵が 40bit と非常に短いこと，ノード全体で 1 つの鍵を共有していること，ユーザに秘密鍵が知られていること，です。これは現在の暗号通信の常識から考えれば全くお粗末と言わざるをえま

[*28] 「クラック」というのはアクセス制限の抜け穴さがしを指す一般的な表現です。ここでのクラックはセキュリティ検証のための，言わば「いい意味」でのクラックです。

[*29] http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html 参照。

せん。

無線 LAN については認証方式を含む新しい規格（IEEE802.1x）が提案されていて，2001 年内には確定するものと思われます。更に一部のベンダでは IEEE802.1x に EAP（Extensible Authentication Protocol: 拡張可能認証プロトコル[*30]）および RADIUS（Remote Authentication Dial-In User Service[*31]）を組み合わせ，認証情報を一元管理できるようにしています。

### 5.7.3 DMCA

米国は WIPO（世界知的所有権機関）の著作権条約を実施するための法案として「DMCA」（Digital Millennium Copyright Act.）を成立させました。これは別名「プロテクト破り規制法案」とも呼ばれていて，「アクセス制限の抜け穴さがしは全面禁止せよ」「その技術を公開する者も処罰せよ」という危険な内容を含んでいます[*32]。

どんな暗号も完全無欠ではありません。時代によりコンピュータのスペックも変わりますし，新たな暗号破りのアルゴリズムが発見されるかもしれません。したがって，暗号のアルゴリズムや運用モデルは常に公開され多くの人の「目」で合法的に検証される必要があります。（「抜け穴探し」自体を違法にしても悪意のクラッカーには（もともと非合法でやってるので）関係ありませんし，一度「世間」に暴かれた「事実」は二度と隠すことができないからです）

## 6 最後に

暗号通信や認証について駆け足で紹介していきました。暗号通信は決して単独の技術ではなく，1章で示した 4 つの条件を満たすための複数の技術の集積です。更に，個々の技術要件を満たしていても運用によって簡単に「穴」が開いてしまう事も事例により示せたと思います。暗号通信システムを考える際に本テキストが多少なりとも参考になれば幸いです。

なお，参考文献を末尾に挙げていますが，インターネット上でもいくつか情報があります。主なものを以下に挙げます。是非参考にしてください。

- 「Federal Information Processing Standards Publications」（http://www.itl.nist.gov/fipspubs/）
- 「信用モデル「信用の輪（web of trust）」のひみつ」（http://pgp.iijlab.net/pgp/trust.html）
- 「PKI 関連技術情報」（http://www.ipa.go.jp/security/pki/pki.html）
- 「OpenPKSD」（http://openpksd.org/）
- 「Kerberos Notes for Japanese」（http://www.rccm.co.jp/~juk/krb/）
- 「DVDCA と大きな嘘」（http://www1.neweb.ne.jp/wa/yamdas/column/technique/esr-dvdj.html）

## 変更履歴

**第 1 版 2001 年 9 月 23 日**　公開第 1 版。

---

[*30] RFC 2284: http://www.ietf.org/rfc/rfc2284.txt 参照。

[*31] http://www.dtc.co.jp/Radius/ 参照。なお Microsoft では IAS（Internet Authentication Service）という製品名で呼ばれています。

[*32] http://www-vacia.media.is.tohoku.ac.jp/~s-yamane/articles/crypto/circumvention.html 参照。

**第 1 版（Rev. 2） 2001 年 10 月 1 日**　IEEE802.1x についての記述を追加した。

**第 2 版 2002 年 2 月 16 日**　章構成を一部変更し社内版を廃止し公開版と統合した。AES,DSA の記述を最新の情報に修正。また PGP,PKI,SSL などに関する記述も修正した。

**第 2 版（Rev. 2） 2002 年 3 月 24 日**　SSL/TLS の章で認証の説明にウソを書いてました。ゴメンなさい。

## 参考文献


[1] Tom Austin. PKI 公開鍵基盤 電子署名法時代のセキュリティ入門. 日経 BP 企画, 2001. 株式会社ニューコム 訳.

[2] Steve Burnett and Stephan Paine. RSA セキュリティ オフィシャルガイド 暗号化. 株式会社 翔泳社, 2002. RSA セキュリティ株式会社 監修，株式会社スリー・エー・システムズ 訳.

[3] Simson Garfinkel. PGP 暗号メールと電子署名. 株式会社 オライリー・ジャパン, 1996. 山本和彦 監訳，株式会社ユニテック 訳.

[4] Charlie Scott, Paul Wolfe, and Mike Erwin. *VPN*. 株式会社 オライリー・ジャパン, 2nd edition, 2000. 歌代和正 監訳，須田隆久 訳.

[5] 新山祐介, 春山征吾. OpenSSH セキュリティ管理ガイド. 株式会社 秀和システム, 2001.