暗号化 / 認証技術とその応用

稲村 雄（日本ベリサイン（株）マーケティング部）

1999 年 12 月 14 日

Internet Week 99　パシフィコ横浜

（社）日本ネットワークインフォメーションセンター編

目次

1 概要

この講演では、次の事柄について説明します。

- 暗号化技術とは何であり、どのようなものがあるのか
- 暗号化技術を応用した認証技術には、どのようなものがあるのか
- 各種セキュリティプロトコル内で、暗号化技術や認証技術がどのように利用されているのか

2 暗号化技術の概説

暗号通信の歴史は、日本ではジュリアスシーザーという呼び名で知られている、共和政ローマ末期の政治家であり軍人であり文学者であるユリウスカエサルが、知人宛の親書を託す使者を信頼できなかったときに始まったとされています。このときカエサルが利用した暗号は、カエサル暗号やカエサルローテーションと呼ばれている、現在ではほとんど実用的ではない単純な換字式暗号でした。

ただし、このような信頼のおけない通信路を介して、いかに安全に通信するかということは、2000 年にも及ぶ難題と言えるでしょう。そして、現在のインターネットでは、その成立自体が研究者向けで性善説に基づいたものであり、接続ホスト数が指数関数的に増加し、誰も全体像を把握できなくなっていることから、いかに安全に通信するかという難題を解決しなければならなくなっています。

暗号化技術は、次のように発達し続け、それに伴って、公開される情報も増加してきました。

1. 通信の存在自体を秘密にする。
2. 手順である「暗号化アルゴリズム」を秘密にする。
3. パラメータである「鍵」を秘密にし、受け手によって異なる鍵を使う。
4. 復号鍵のみを秘密にする（1976 年）。
5. 鍵情報を中立機関に供託する（1993 年）。

このように、本来は情報を秘匿するための技術である暗号化技術は、逆説的にも、より多くの情報を公開することが可能なシステムを模索するという形で発達してきたわけです。

ここでは、まず、暗号化技術を定義し、対称アルゴリズムと非対称アルゴリズムについて説明します。

2.1 定義

暗号化技術は、『通信文など（＝ 平文）を第三者には意味不明な形（＝ 暗号文）に変換することで、当事者以外にとっての有用性を失わせしめるための技術』と定義できます。つまり、暗号化技術とは、当事者にとってのみ可逆なデータ変換技術となります。

そして、その変換方法の数は、たとえば平文空間が N ビットであったときには 2^N！通りとなります。言い換えれば、2^N！通りの変換パターンの一覧が「暗号化アルゴリズム」であり、どのパターンによって変換されたかを示すパラメータが「鍵」となります。ただし、N の値が大きくなるに従って、利用できる変換パターン数と鍵数が膨大なものとなっていきます。このため、実際の暗号化技術は、利用できる変換パターンから、解読しづらいものをいかに取り出して利用するかということになります。

2.2 対称アルゴリズム

対称アルゴリズムは、暗号化と復号に同一の鍵を使うアルゴリズムで、秘密鍵暗号や共通鍵暗号とも呼ばれています。また、1976 年までは、暗号化アルゴリズムと言えばこの方式のみであったため、慣用暗号とも呼ばれています。

対称アルゴリズムでは、あらかじめ何らかの方法を使って、送信者と受信者が共通の鍵を持つようにします。そして、送信者は、その鍵を使って平文を暗号化した後、受信者に渡します。受信者は、受け取った暗号文を送信者と同一の鍵を使って復号します。

このような対称アルゴリズムでは、Shannon の情報理論に基づき、混乱（Confusion）のための置換と拡散（Diffusion）のための転置という 2 つの操作が実行されます。これらの操作を効果的に実現している方式に、SPN（Substitution Permutation Network）と呼ばれるものがあります。SPN では、鍵と平文／暗号文との間で複雑な演算が実行され、暗号化や復号が実現されています。

また、実用的ではありませんが、通信するメッセージと等長な真の乱数鍵を利用する Onetime Pad と呼ばれる方法は、絶対に破られない暗号となることが数学的に証明されています。ただし、現実には、メッセージと等長の乱数鍵を安全に受け渡すことは困難だと思えます。

このような Onetime Pad での問題点も含め、対称アルゴリズムには次の 2 つの課題が存在しています。

- 鍵の安全な共有方法が必要であること
- 通信する相手ごとに異なる鍵が必要であること

また、対称アルゴリズムによる処理方法は、次の 2 種類に大別されます。

- ストリーム暗号（2.2.1 を参照）
- ブロック暗号（2.2.2 を参照）

2.2.1 ストリーム暗号

対称アルゴリズムによるストリーム暗号では、平文や暗号文を先頭から順番に処理していきます。このときの処理単位は、基本的にはビット単位となりますが、利用される計算機が自然に扱えるサイズを反映して、バイト単位やワード単位で処理されることもあります。

実際にストリーム暗号を利用するときには、送信者と受信者が共有している 100 ビット程度の鍵からメッセージと等長の疑似乱数列を生成し、その乱数列を使って暗号化や復号を実施します。つまり、ストリーム暗号は、絶対に破ることができない Onetime Pad の簡易版と考えることができます。ただし、利用する乱数列が 100 ビット程度の鍵から生成されるため、Onetime Pad ほどの安全性は確保されません。

代表的なストリーム暗号には、RC4、SEAL、WAKE があります。また、ストリーム暗号は、軍事暗号に利用されているとも言われています。これは、ストリーム暗号が安全性を評価しやすいためです。

2.2.2 ブロック暗号

対称アルゴリズムによるブロック暗号では、平文や暗号文が 64 ビットや 128 ビットの一定サイズのブロックに分割された後に、ブロックごとに処理されます。このとき、最後のブロックには、他のブロックとサイズを合わせるためにパディングが実施されるため、平文よりも暗号文のブロックサイズが大きくなる傾向があります。

もっとも有名なブロック暗号に DES（Data Encryption Standard）があります。DES は、米国 NBS（現在の NIST）の公募に対する IBM 社からの提案に基づいた暗号化アルゴリズムで、1976 年以来、標準的な暗号方式として利用されてきています。ただし、鍵長が 56 ビットと短いため、現在ではほぼ寿命が尽きたと言われています。

このような理由から米国 NIST は、DES に代わるアルゴリズムとして AES（Advanced Encryption Standard）を 1997 年 9 月から公募し始めています。AES については *http://csrc.nist.gov/encryption/aes/aes_home.htm* から詳細を知ることができますが、最低条件として次の 3 点を満たす必要があります。

- 対称鍵暗号であること
- ブロック暗号であること

- 鍵長とブロック長の組み合わせとして、128 ビットと 128 ビット、192 ビットと 128 ビット、256 ビットと 128 ビットに対応すること

この AES に対する候補アルゴリズムとしては、次の 15 種類が提案され、このうちの MARS、RC6、RIJNDAEL、SERPENT、TWOFISH の 5 つが、1999 年夏に第 2 次選考の候補となっています。

- CAST-256
- CRYPTON
- DEAL
- DFC
- E2
- FROG
- HPC
- LOKI97
- MAGENTA
- MARS
- RC6
- RIJNDAEL
- SAFER+
- SERPENT
- TWOFISH

2.3 非対称アルゴリズム

非対称アルゴリズムは、公式には 1976 年に W. Diffie と M. Hellman によって考案されたものとされています。ただし、一部では、1966 年に米国 NSA で利用されていたという説や、1970 年に英国 CESG で利用されていたという説もあります。

非対称アルゴリズムの基本は、落とし戸（Trapdoor）付き一方向関数です。この関数は、片方向への計算は容易ですが、逆方向への計算は非常に困難なものとなっています。ただし、Trapdoor という特別な知識を持つ者は、この逆方向の計算を実行することができます。このような一方向関数の性質を利用することで、誰もが容易に暗号化できるが、特定の者しか復号できない暗号化システムになると考えられています。

このような非対称アルゴリズムの先駆けとして、R. Markle による提案（図 1）があります。

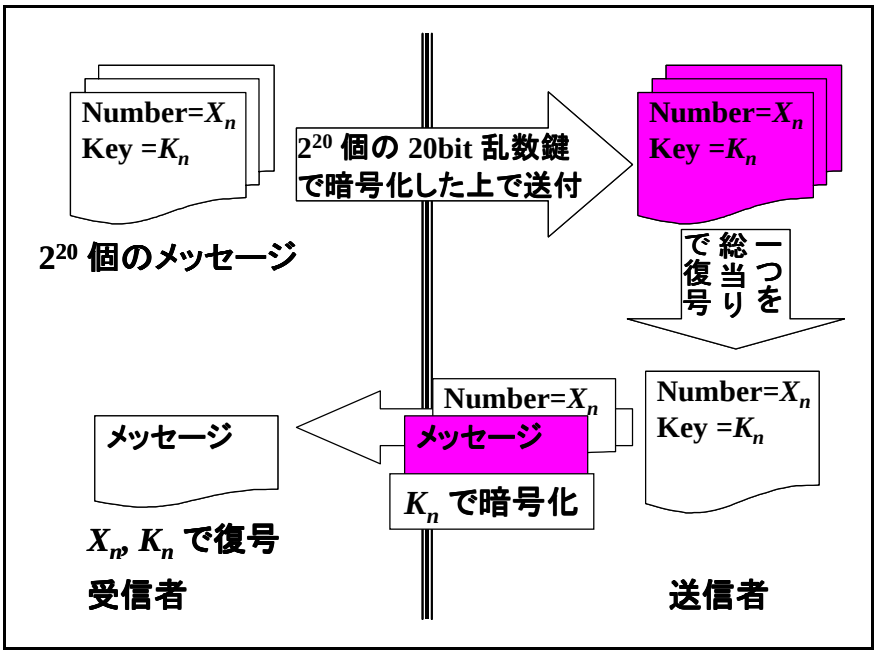

図 1：R. Markle による提案内容

この提案では、受信者は鍵番号と鍵自体を記録した2^{20}個のメッセージを用意し、2^{20}個の 20 ビット乱数鍵で暗号化し、送信者に渡します。送信者は、受け取った暗号文のどれか 1 つを、20 ビットの可能な鍵パターンすべてを試みるという総当り方式で復号し、中に記録された 1 つの番号と鍵とを入手します。そして、その鍵を使って送信メッセージを暗号化し、鍵番号とともに受信者に送り返します。受信者は、受け取った鍵番号に対応した鍵によってメッセージを復号します。このような処理によって、受け渡されるメッセージが第三者に搾取されたとしても、その第三者は手に入れた、2^{20}個のメッセージすべてを総当たりで復号しなければならないため、通信の安全性が確保されます。

この方法には、Trapdoor となる鍵情報が2^{20}個という膨大な数である、送信者の手間がかかりすぎるなどの問題点はありますが、この後に示す公開鍵暗号の概念を的確に表していると思います。

2.3.1 公開鍵配布系

非対称アルゴリズムを利用した公開鍵配布系は、対称アルゴリズムに対する補助的な役割を果たしています。公開鍵配布系では、各ユーザが公開している情報から対称暗号方式で使用する共通鍵を生成します。つまり、送信者は受信者の公開情報を基に暗号化し、受信者は送信者の公開情報を基に復号します。

このような公開鍵配布系では、暗号化や復号のために送受信者の公開情報によって生成される共通鍵が同一のものとなり、それぞれの鍵を生成できるのが送受信者のみとならなければなりません。これは、送信者と受信者が互いに秘密の情報を持ち、それを公開情報と組み合わせることで実現されています。

公開鍵配布系の代表的なアルゴリズムには、Diffie-Hellman 方式があります。Diffie-Hellman 方式は、W. Diffie と M. Hellman によって 1976 年に考案された世界初の非対称アルゴリズムであり、離散対数問題の困難性に基づいたものとなっています。

2.3.2 公開鍵暗号

非対称アルゴリズムによる公開鍵暗号も、W. Diffie と M. Hellman によって 1976 年に考案されました。公開鍵暗号では、暗号化と復号に異なる鍵が利用されます。

まず、受信者は、数学的に特殊な関係となる、暗号化のための鍵と復号のための鍵をそれぞれ生成し、暗号化のための鍵のみを公開します。送信者は、受信者によって公開されている暗号化鍵を使って平文を暗号化し、その内容を受信者に送ります。受信者は、手元にある復号鍵によって暗号文を復号することで、内容を参照します。このように、公開鍵暗号では、一方の鍵で暗号化されたデータは、もう一方の鍵でしか復号できません。また、一方の鍵のみから残りのもう一方の鍵を導き出すことは非常に困難なものとなっています。

このような公開鍵暗号によって、先ほど示した対称アルゴリズムに対する次の 2 つの課題が解決されます。

- 鍵の安全な共有方法が必要であること
- 通信する相手ごとに異なる鍵が必要であること

まず、1 番目の課題は、暗号化鍵を公開してしまうことで解決されます。また、2 番目の課題は、すべての送信者に対して 1 対の暗号化鍵と復号鍵だけを利用できるため、問題とはならなくなります。このようなことから、公開鍵暗号は、インターネット環境で利用するためには必須の技術と言えるでしょう。

もっとも有名な公開鍵暗号方式に RSA があります。RSA は、R. Rivest、A. Shamir、L. Adelman の 3 名によって 1978 年に発明された暗号化アルゴリズムです。RSA は、非常に大きな数の素因数分解の困難性に基づいたもので、公開鍵暗号のデファクトスタンダードとなっています。

また、他の非対称アルゴリズムとしては、1985 年に、N. Koblitz と V. S. Miller という 2 名のカナダ人がそれぞれ個別に考案した楕円曲線暗号があります。この暗号方式は、新たな方式ではなく Diffie-Hellman 方式などの既存システムを楕円曲線上に実装したものです。現状では十分な検査が終了していませんが、他の非対称アルゴリズムと比較して短い鍵長で安全性を確保できると考えられているため、将来 RSA に代わる可能性があります。

3 認証技術の発展

ネットワーク上の人物を特定するための電子認証技術は、基本的に実世界での認証と同じように、物理的な特徴や所有物をあらかじめ登録しておいたデータと照合して確認するものとなります。ただし、電子データは容易に複製できてしまうため、実世界での認証方法を単純に適用することはできません。

ECOM（電子商取引実証推進協議会）によるガイドラインでは、認証手段を次のように層別しています。

- 指紋、網膜、声紋、筆跡などの本人固有の情報による認証
- クレジットカード、運転免許証などの所持品による認証
- パスワード、暗証番号、デジタル署名、公開鍵証明書などの秘密情報による認証

このうち 1 番目と 2 番目の方法は、入退室管理などの用途には利用できますが、ネットワーク経由の認証には利用できません。したがって、ネットワーク経由の認証には、3 番目の方法による認証手段を利用することになります。このような認証手段には、次の 5 種類があります。

- パスワード認証（3.1 を参照）
- チャレンジ & レスポンス（3.2 を参照）
- ワンタイムパスワード（3.3 を参照）
- Kerberos（3.4 を参照）
- 電子署名（3.5 を参照）

3.1 パスワード認証

パスワード認証には、次の 2 つの方式があります。

- 単純パスワード方式
- 暗号化パスワード方式

単純パスワード方式では、ユーザとシステムが同一のパスワードを共有し、ユーザが送出したパスワードをシステムで照合します。これに対して、暗号化パスワード方式では、ユーザのパスワードは、暗号化してシステム上のデータベースに保存されます。そして、ユーザが送出したパスワードは、システムによって暗号化された後に照合されます。

このような方法によるパスワード認証では、パスワード自体が直接受け渡されます。そして、このようなパスワードの受け渡しは、インターネット環境でも利用されています。ただし、インターネット上では、受け渡されているパスワードを簡単に盗み見ることができてしまいます。このため、パスワード認証は、インターネット経由ではなく、直接システムに接続していなければ安全性を確保できません。

3.2 チャレンジ & レスポンス

チャレンジ & レスポンスでは、任意のデータからそのデータ特有とみなせる百数十ビット程度の短い情報を抽出する、メッセージダイジェストという技術が利用されます。メッセージダイジェストでは、逆演算が不可能な暗号学的一方向ハッシュ関数が利用されます。暗号学的一方向ハッシュ関数には次のような特性があるため、メッセージダイジェストを元データの「指紋」として扱うことができます。

- 元データが1ビット異なっただけでも、メッセージダイジェストはまったく異なったものとなる。
- 特定のメッセージダイジェストに対応する元データを見つけ出すことが非常に困難である。
- 同一のメッセージダイジェストを得られるような 2 種類のメッセージを見つけ出すことが非常に困難である。

メッセージダイジェストを抽出するための代表的なアルゴリズムには、MD2、MD4、MD5 と、SHA-1 があります。MD2、MD4、MD5 は、R. Rivest によって開発されたアルゴリズムで、128 ビット長のメッセージダイジェストが抽出されます。また、SHA-1 は、米国 NIST が NSA とともに開発したアルゴリズムで、160 ビット長のメッセージダイジェストが抽出されます。

チャレンジ & レスポンスでは、ユーザとシステム間で同一のパスワードが共有されます。そして、ユーザ認証が必要となったときには、システムからユーザにチャレンジデータという 1 回かぎりのデータが送出されます。ユーザは、チャレンジデータと自分のパスワードを組み合わせ、一方向関数によってメッセージダイジェストを生成し、その内容をシステムに返します。システムは、チャレンジデータの送出と同時に自らもメッセージダイジェストを生成し、ユーザから返されたメッセージダイジェストと照合することで、認証の可否を決定します。この方法では、パスワード自体はネットワーク上でいっさい受け渡されません。

3.3 ワンタイムパスワード

S/Key などのソフトウェア方式によるワンタイムパスワードでは、システム側がチャレンジデータを、そして、ユーザ側がパスワードをあらかじめ用意しておきます。そして、ユーザは自分のパスワードとチャレンジデータを組み合わせ、一方向関数によって複数のワンタイムパスワードを生成し、最後に生成した内容のみをシステムに登録しておきます。そして、実際の認証では、システムがチャレンジデータとともに N 番目のワンタイムパスワードをユーザに要求し、そのワンタイムパスワードをユーザ側で計算して返します。システムは、返された N 番目のワンタイムパスワードから N ＋ 1 番目のワンタイムパスワードを計算し、前回のデータと照合して認証の可否を決定した後、次回の認証に備えてその値を保存します。

このような方法で認証を繰り返し、あらかじめ用意したワンタイムパスワード列をすべて使い切ったときには、再度ユーザのパスワードとチャレンジデータを組み合わせて、複数のワンタイムパスワードを生成します。このように、ワンタイムパスワードでは、これまでに示したパスワード認証やチャレンジ & レスポンスとは異なり、ユーザのパスワード自体をシステムに登録する必要がありません。

3.4 Kerberos

Kerberos では、共通鍵暗号のみを使うことで、図 2 に示すように、データ暗号化だけでなく認証も実現しています。

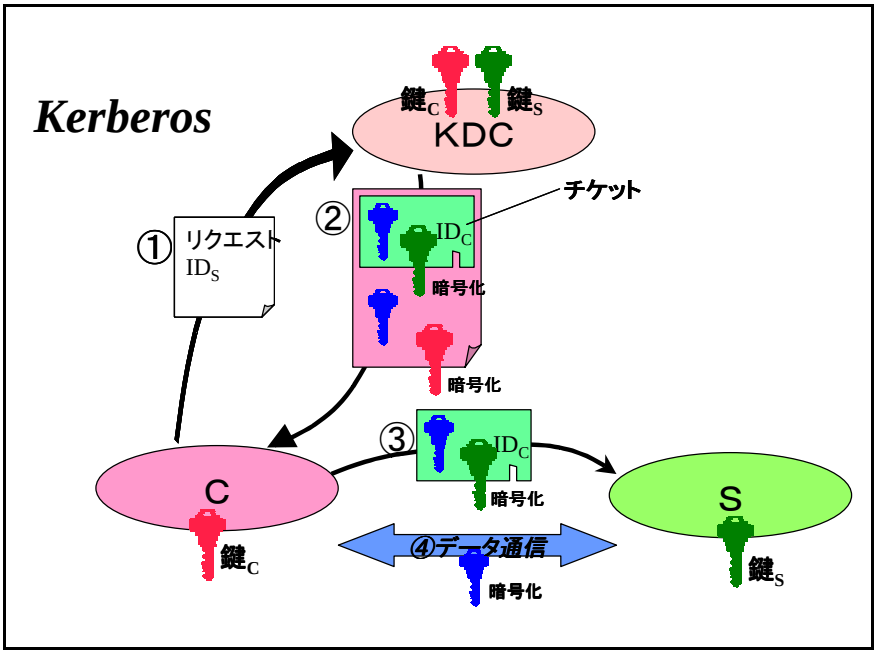

図 2：Kerberos による認証

Kerberos では、あらかじめ各利用ユーザのそれぞれの鍵を、KDC（Key Distribution Center）という中央の鍵管理センターにすべて登録しておきます。そして、実際にユーザ A がユーザ B と通信するときには、ユーザ A はユーザ B の識別番号を KDC に通知します。KDC は、セッション鍵を生成した後、ユーザ B の鍵によって暗号化したセッション鍵（＝チケット）と、ユーザ A が利用するセッション鍵を、ユーザ A の鍵で暗号化して返します。ユーザ A は、自らの鍵で自分用のセッション鍵とチケットを復号し、取り出したチケットをユーザ B に渡します。ユーザ B は、自らの鍵でチケットを復号することでセッション鍵を入手します。ここまでの処理によって、ユーザ A とユーザ B は、互いが入手したセッション鍵を使って安全にデータを通信できるようになります。

3.5 電子署名

これまでに示した認証技術には、いくつかの問題点があります。まず、パスワード認証では、秘密情報であるパスワードが直接ネットワーク上で受け渡されてしまいます。チャレンジ & レスポンスでは、システム内に保管されるパスワードが漏洩する恐れがあります。ワンタイムパスワードを利用することで安全性は高いものとなりますが、認証のみが保証され、その後のデータの受け渡しは保護されません。Kerberos では、すべての通信に KDC という中央の鍵管理センターが必要となります。そして、いずれの認証技術も、電子商取引で重要となる「否認」を防止することができません。

このような問題は、電子署名によって解決されます。電子署名は、実世界での署名や印鑑押捺などと同様に、内容が改ざんされていないことを保証するために作成者がデータに付加する「印」で、デジタル署名とも呼ばれています。通常、電子署名は、メッセージダイジェストと非対称暗号技術の併用によって実現されています。

まず、データ作成者は、データから一方向関数によってメッセージダイジェストを生成し、自分だけが所持している秘密鍵で暗号化して署名データを作成します。データを検証したいときには、受け取ったデータから一方向関数によってメッセージダイジェストを生成し、データとともに送られてきた署名データをデータ作成者の公開鍵で復号して両方のメッセージダイジェストを照合します。

また、電子署名を用いた認証では、最初にシステムが認証のためにユーザにチャレンジデータを送出します。ユーザは、自分の秘密鍵を使って電子署名を作成し、その署名をシステムに返します。システムは、受け取った電子署名をユーザの公開鍵を使って復号し、元のチャレンジデータと照合することで認証の可否を決定します。

4　実プロトコルでの利用形態

ここでは、次に示す各種セキュリティプロトコル内で、暗号化技術や認証技術がどのように利用されているかを説明します。

- SSH（4.1 を参照）
- S/MIME（4.2 を参照）
- SSL（4.3 を参照）
- IPSec（4.4 を参照）

4.1 SSH

SSH（Secure SHell）は、rlogin や rsh などの BSD 系 UNIX で使われる rcmd 機能の置き換えとして、フィンランドの Tatu Ylonen によって考案されたセキュリティプロトコルです。SSH では、サーバとクライアントで同一のユーザアカウントが存在していることが前提とされ、対称アルゴリズムと非対称アルゴリズムが併用されています。また、バージョン 2.0 以降の SSH では有償となる商業目的利用の範囲が拡大されたため、現状でもバージョン 1.2 が多数利用されているようです。そのため、ここでは、そのバージョン 1.2 系列で利用されるプロトコルを説明します。

SSH では、図 3 に示す手順によってセッション鍵やクライアントの認証が実施されます。そして、これらの処理が終了すると、セッション鍵を利用した共通鍵暗号方式による処理が実施されます。

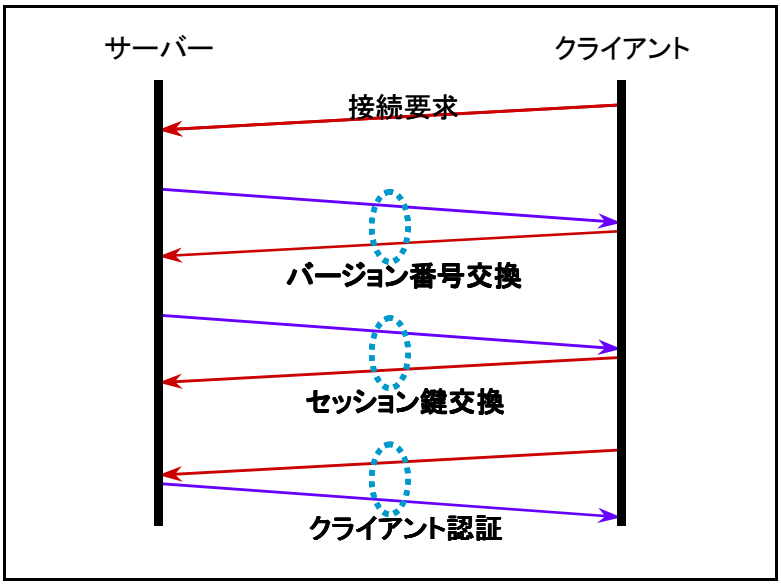

図 3：SSH プロトコル概要図

このうちセッション鍵の交換では、まずサーバの公開鍵がサーバからクライアントに送られます。クライアントは、疑似乱数によってセッション鍵を生成した後、サーバの公開鍵を使って暗号化してサーバに送り返します。そして、サーバは、受け取ったセッション鍵を自分の秘密鍵で復号します。このような処理によって、サーバとクライアントは、一時的に利用するためのセッション鍵をともに入手します。

また、SSH でのクライアントの認証は、次のいずれかの方法となります。

- クライアントホストごとの公開鍵を利用してホスト単位で認証するホスト RSA 認証
- ユーザごとの公開鍵を利用してユーザ単位で認証するユーザRSA認証
- ログインのためのパスワードをセッション鍵で暗号化して利用するパスワード認証

これらの認証方法のうち、ユーザ RSA 認証は、次のような手順で実施されます。

1. クライアントからサーバに対して、ユーザ名と公開鍵が渡される。
2. サーバは、ユーザアカウントに基づいて公開鍵を検証した後、チャレンジデータを生成し公開鍵で暗号化してクライアントに送り返す。
3. クライアントは、秘密鍵を使ってチャレンジデータを復号した後、メッセージダイジェストを抽出し、レスポンスデータとしてサーバに送り返す。
4. サーバは、チャレンジデータから生成したメッセージダイジェストとレスポンスデータを照合し、認証の可否を決定する。

このような SSH による認証は、ユーザアカウントの存在が仮定できるため比較的容易に実現されます。ただし、なりすましによる不正アクセスに対処するために、定期的に各ホストの公開鍵を変更する必要があります。

また、SSH を利用することで、X、FTP、SMTP、POP、TELNET などの他のプロトコルに対して安全な通信路を提供したり、VPN（Virtual Private Network）に類似した機能を容易に実現したりすることができます。

4.2 S/MIME

S/MIME（Secure/Multipurpose Internet Mail Extensions）は、RSA 社によって提案されたセキュアメール規格で、電子メールでの暗号化と電子署名処理を提供しています。S/MIME では、ユーザと公開鍵との結び付きを証明書で保証し、SSH と同様に対称アルゴリズムと非対称アルゴリズムを併用しています。従来は PGP が主に利用されていたのですが、1997 年に Netscape Messenger や Microsoft Outlook Express などで採用されたことで S/MIME のシェアが一気に拡大しました。

実際に S/MIME を利用するときには、まず MIME 形式でデータを用意します。そして、そのデータを一時的な秘密鍵（＝セッション鍵）を使って対称方式で暗号化します。また、暗号化のために利用した秘密鍵は、受信者の公開鍵を使って非対称方式で暗号化します。この 2 つを対で利用することで、受信者のみがデータを参照できるようになります。このようにセッション鍵で封印することから、この方法は Digital Envelope と呼ばれています。

また、S/MIME では、署名付きデータとして次の 2 種類の表現形式を利用できます。

- PKCS オリジナル形式である SignedData
- MIME 標準形式である Multipart/signed

このうち SignedData では、元データと署名データを 1 つにまとめて PKCS 形式として構造化します。このため、PKCS と互換性のないツールでは、受け取ったデータを読み取ることができなくなります。これに対して、Multipart/signed では、元データはそのままで署名データが MIME マルチパートとして付加されるため、通常のメーラでも元データだけは読むことができます。

このような S/MIME の利用では、送信者や受信者の公開鍵をどのように入手するかという点を考慮する必要があります。たとえば、メールを暗号化して送信するためには受信者の公開鍵が必要となり、署名付きメールを受け取ったときには送信者の公開鍵が必要となります。このような課題は、デジタル証明書を利用することで解決でき、スケーラブルな相互運用が確保されます。

デジタル証明書は、公開鍵暗号技術の補完的役割を果たし、ある公開鍵の持ち主が本当に申告どおりの人物であるかを保証するための機構です。公開鍵を保証する代表的な方法には、次の 2 つがあります。

- 草の根的解決（PGP）
- 信頼された第三者機関による保証

草の根的解決は、「友達の友達は友達」方式とも呼ばれています。つまり、信頼関係にある二者の一方が、ある公開鍵の持ち主を電子署名によって保証することで、他方は信頼関係に基づいてその公開鍵の持ち主を信頼します。

これに対して、信頼ある第三者機関による保証では、認証局（CA：Certificate Authority）によって公開鍵とその鍵の持ち主の結び付きが証明されるため、誰もが公開鍵の持ち主を検証できます。また、その内容は、姓名、所属、電子メールアドレスなどのユーザの素性と公開鍵データに対して、認証局自らの電子署名が施されたものとなります。現在では、ITU-T 勧告 X.509 とその拡張で定められた仕様に基づく、図 4 のような公開鍵証明書が主流となっています。

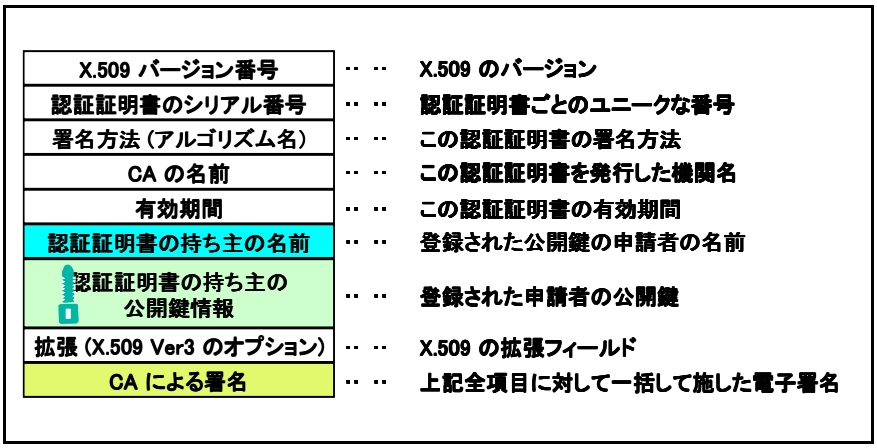

図 4：X.509 での証明書内容

4.3 SSL

SSL（Secure Sockets Layer）は、Netscape Communications 社が提案したセキュリティプロトコルです。SSH と同様に SSL も、クライアント／サーバモデルでの利用が前提となっています。一般にクライアント／サーバモデルでは、認証用データをあらかじめ用意しておくことが不可能です。このため、SSL では、デジタル証明書を利用して相互運用性を確保し、対称アルゴリズムと非対称アルゴリズムを併用しています。また、Netscape Communications 社によって提案されたため、現在のところ SSL は主に WWW システムで利用されています。

SSL では、図 5 に示すような手順によって、クライアントとサーバ間で暗号化アルゴリズムが決定され、セッション鍵が生成され、サーバとクライアントが互いを認証しています。

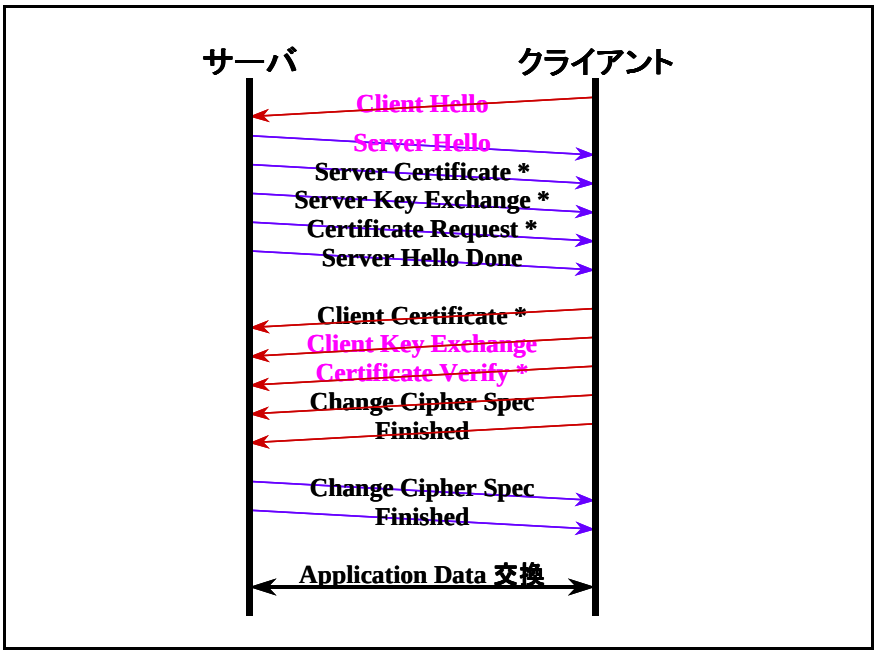

図 5：SSL プロトコル概要図

ここでは、最初にクライアントが Client Hello を送り、それに対してサーバが Server Hello を返すことでセッションが開始されます。次に、サーバは、自らの証明書である Server Certificate をクライアントに送った後、Server Hello Done によって送信の終了を告げます。クライアントが Client Key Exchange を送ることで、暗号化などのためのネゴシエーションが終了します。そして、クライアントとサーバが互いに Change Cipher Spec と Finished を送り、設定した暗号化アルゴリズムやセッション鍵などを利用し始めます。

また、このようなクライアントとサーバ間でのやりとりでは、クライアントに対する認証のために、Certificate Request、Client Certificate、Certificate Verify というオプションが利用されることもあります。

このようなやりとりのうち最初に実施される Client Hello と Server Hello では、図 6 に示すメッセージが受け渡されることで、暗号化とメッセージダイジェストのアルゴリズムが決定されます。

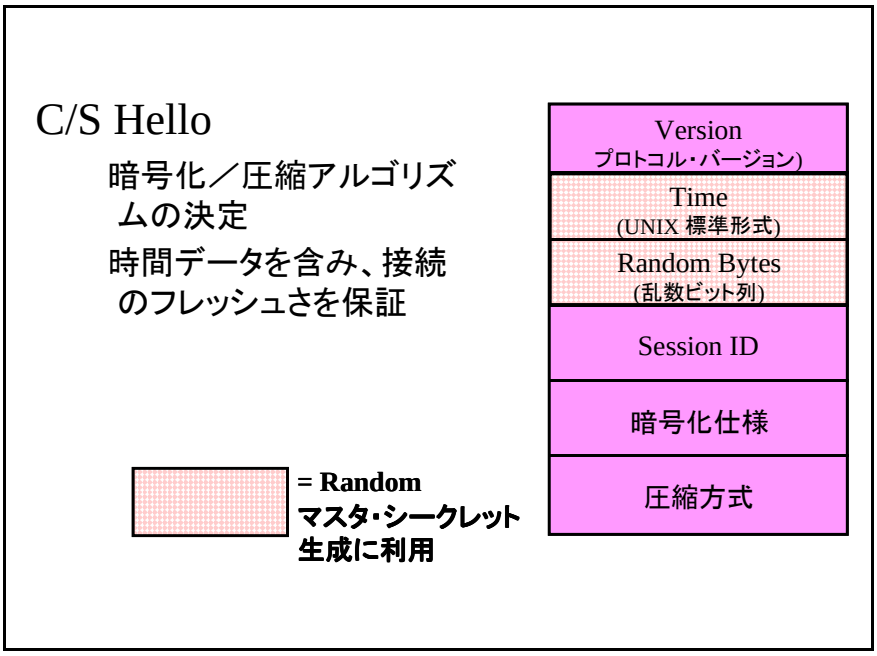

図 6：Client Hello と Server Hello でのメッセージ内容

また、これらのメッセージには時間データが含まれているために、接続の鮮度が保証されています。さらに、サーバとクライアントがともに疑似乱数を生成し利用することで、安全性も高められています。

クライアントから送られる Client Key Exchange は、このような接続の確立のなかでもっとも重要な部分です。Client Key Exchange では、セッション鍵などを生成するためのマスタシークレットと呼ばれるデータの元となるデータ（= プリマスタ・シークレット）がサーバに送出されています。RSA を利用したときには、このデータは、プロトコルバージョンを表す 2 バイトと乱数ビット列 46 バイトを合わせた 48 バイトで構成され、サーバの公開鍵によって暗号化されます。サーバでは、クライアントから受け取った Client Key Exchange、Client Hello と、自らの Server Hello の乱数ビット列を基に、SHA-1 と MD5 の 2 種類の一方向関数アルゴリズムを使って、クライアントとサーバのみが知り得る 48 バイトのマスタシークレットを生成します。そして、SSL では、このマスタシークレットを

使って、暗号化鍵、初期化ベクタ、MAC 計算用秘密データなどに利用するためのキーブロックも生成されます。

SSL では、SSH とは異なりユーザアカウントの存在が仮定できないため、認証のために証明書の存在が必須となっています。ただし、このような証明書の利用によって、なりすましによる不正アクセスが防止されています。

また、暗号化技術は、各国の政策の影響を大きく受けるため、高強度な暗号化技術を米国外で利用することが困難なことがあります。図 7 に、1999 年 11 月 1 日現在での米国輸出管理規制の内容を示します。

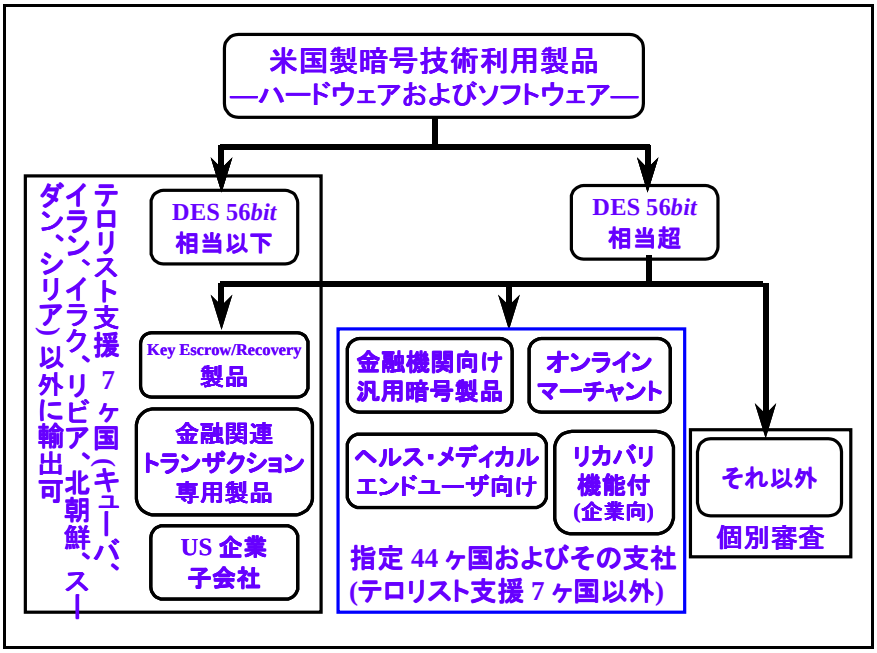

図 7：米国輸出管理規制（1999 年 11 月 1 日現在）

SSL では、暗号化の強度を変更した 2 種類のバージョンを用意することで米国輸出管理規制を満たしているだけでなく、グローバルサーバ ID によって米国外で高強度な暗号を利用する手段も提供しています。

グローバルサーバ ID を利用するサーバにアクセスしたときには、クライアントとサーバ間で受け渡される Client Hello と Server Hello の後に、グローバルサーバ ID を含んだ Server Certificate がサーバからクライアントに送られます。グローバルサーバ ID に対応しているバージョン 4 以降の Netscape NavigatorやInternet Explorerをクライアントとしていたときには、クライアントからサーバに対して Reset が送られた後に、128 ビット長で暗号化された Client Hello と Server Hello によるセッションが再開されます。

4.4 IPSec

IPSec（Security Architecture on Internet Protocol）は、IETF（Internet Engineering Task Force：インターネット技術検討部会）が中心となって標準化を実施しているセキュリティプロトコルです。IPSec では、SSH や SSL のように通信ごとに認証を実施することが現実的ではないため、すべての通信を始める前の段階で、暗号化方式やそこで用いる鍵などのパラメータをあらかじめ決定しておくような仕様となっています。

また、IPSec では、暗号化などのためのパラメータの調整が、通信する二者間であらかじめ完了しているときに、データの暗号化や完全性が保証されます。このようなことから、他のセキュリティプロトコルでは鍵管理方式との併用が必要でしたが、IPSec では鍵管理方式は分離され、手動配布、Diffie-Hellman、Kerberos などの任意の方式が利用できるようになっています。

このような鍵管理方式の柔軟性は、SA（Security Association）と SPI（Security Parameters Index）によって実現されます。SA は、暗号化や認証のためにプロセス間で必要となるパラメータの集合です。鍵管理方式によってあらかじめ SA を設定しておくことで、IPSec ではその SA を使用して暗号化や認証が実施されます。また、SPI は、複数の SA から特定のものを選び出すためのインデックスとして利用されます。

IPSec では、このような方法によって任意の鍵管理方式が利用できます。そして、鍵管理方式を自動化するシステムの 1 つに IKE（Internet Key Exchange）があります。IKE は、通信相手との間で利用する暗号化アルゴリズムや鍵などのパラメータを定めるためのプロトコルであり、Diffie-Hellman 方式を基に安全に鍵を交換できます。

また、IPSec では、IP パケットの完全性を保証するための AH（Authentication Header）と、IP パケットを暗号化して送付するための ESP（Encapsulating Security Payload）が付加されています。ただし、最近では、完全性を保証するための機構が ESP に付加されたため、今後は ESP のみが利用され AH は利用されなくなる可能性があります。

さらに、IPSec のオペレーションモードには、次の 2 種類があります。

- トンネルモード
- トランスポートモード

トンネルモードでは、完全な IP パケットであるペイロードと IP ヘッダが対象となり、IPSec による操作後に新たに IP ヘッダが付加されます。これに対してトランスポートモードでは、トランスポート層のデータが対象となり、IPSec による操作後に初めて IP ヘッダが付加されます。このような違いによって、トンネルモードではデータ全体が大きくなりますが、オリジナルのヘッダ情報までもが保護されています。

このように IPSec では、コネクションレスという IP の特性から鍵交換と実運用が完全に分離されています。このため、小規模な LAN 環境においては手作業でパラメータを設定し、大規模なエクストラネットなどでは証明書を利用して相互運用性を確保するというように、利用環境に応じた柔軟な構成が可能となっています。