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）

1998 年 12 月 15 日

Internet Week 98　国立京都国際会館

（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 1
2　目的 1
3　暗号化技術の概説 1
4　電子認証技術 8
5　公開鍵証明書と公開鍵基盤 11
6　セキュリティプロトコル 13

1　概要

現在、インターネット環境上で利用する暗号技術として、非対称アルゴリズムによる公開鍵暗号が最適なものとなっています。また、認証方式としては、電子商取引などで重要となる否認も防止できる公開鍵証明書方式が有効なものとなっています。さらに、現状では、S/MIME、SET、SSL、TLS、IPSec などのセキュリティプロトコルが標準化されたり標準化作業中ですが、将来は IP バックボーン上に PKI（Public Key Infrastructure）バックボーンが構築されることで、あらゆる認証、秘匿性、安全性、否認防止が統合されていくと思われます。

2　目的

ここでは、公開鍵暗号と公開鍵証明書を中心に暗号化と認証技術についての概念を体系的に説明します。また、現在利用されていたり標準化作業中である、さまざまなセキュリティプロトコルについても説明します。

3　暗号化技術の概説

暗号通信は、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、知人に宛てた手紙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、カエサル暗号と呼ばれている、現在ではほとんど実用的でない単純な換字式（かえじしき）暗号でした。

このように、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も、インターネット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、その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. 暗号化アルゴリズムを秘密にする
3. 「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）

ここでは、暗号技術を定義し、暗号技術の対称アルゴリズムと非対称アルゴリズム、暗号アルゴリズムの安全性と、電子署名について説明します。

3.1 定義

暗号技術は、『通信文など（= 平文（ひらぶん））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、暗号技術とは、可逆なデータ変換技術です。このことから、平文空間が N ビットであったときには、その変換方法は$2^N!$通りとなります。そして、どのパターンによって変換されたかを示すのが「鍵」となります。つまり、暗号技術とは、このような変換パターンの一覧であると考えることができます。ただし、N の値が大きくなることで、利用できるパターン数と鍵数が膨大なものとなってしまいます。したがって、実際の暗号技術は、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効率的に取り出して利用するかということに尽きると言えるでしょう。

3.2 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、共通鍵暗号、慣用暗号などとも呼ばれ、1976 年までは、暗号と言えば、この方式のみでした。このアルゴリズムでは、Shannon の情報理論に基づき、置換操作による混乱（Confusion）と転置操作による拡散（Diffusion）が、たとえば SPN（Substitution Permutation Netwaork）などの形式をとることにより、効果的に実現されています。

また、一般には実用的ではありませんが、Onetime Pad と呼ばれる方法では、通信するメッセージと等長の乱数鍵を利用することで、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。

このような対称アルゴリズムの基本的な方式は、鍵と平文 / 暗号文との間で複雑な演算を実行することで暗号化や復号を実現するものとなっています。そして、その処理方法は、次の 2 種類に大別されます。

- ストリーム暗号
- ブロック暗号

また、対称アルゴリズムの問題点は、次の 2 点となります。

- 鍵の安全な共有方法が必要
- 通信する相手ごとに異なる鍵が必要

3.2.1 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ます。ただし、その処理単位は、ビット、バイト、ワードなどさまざまなものとなっています。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鍵から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。

ストリーム暗号アルゴリズムの代表例には、RC4、SEAL、WAKE があります。このうち、RC4 は、RSA の開発者のひとりである Ron Rivest によって開発された、可変長鍵が利用できるストリーム暗号であり、Netscape Navigator や Internet Explorer で利用されている SSL（Secure Sockets Layer）のデフォルトアルゴリズムとして知られています。さらに、RC4 は、非公開のアルゴリズムなのですが、リバースエンジニアリングされてしまい、そのソースコードやアルゴリズムは、事実上公開されてしまっています。

3.2.2 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、平文や暗号文を一定サイズのブロックに分割した後、処理します。

ブロック暗号の代表的な利用例に、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、そろそろ寿命が尽きつつあります。

このような理由から米国 NIST は、1997 年 9 月から DES に代わるアルゴリズムとして AES（Advanced Encryption Standard）を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 点を満たしている必要があります。

- 対称暗号であること
- ブロック暗号であること
- 鍵長とブロック長の組合せとして、128 ビットと 128 ビット、192 ビットと 128 ビット、256 ビットと 128 ビットの各組合せに対応すること

現在 AES に対する候補アルゴリズムとしては、次のようなものが挙げられています。

- CAST-256
- CRYPTON
- DEAL
- DFC
- E2

- FROG
- HPC
- LOKI97
- MAGENTA
- MARS
- RC6
- RIJNDAEL
- SAFER+
- SERPENT
- TWOFISH

3.3 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 で認識されていたとするものや、1970 年に英国 CESG で認識されていたとの主張もあります。

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。この関数とは、片方向への計算は容易であるが、逆方向への計算は非常に困難なもののことです。ただし、Trapdoor という特別な知識を持つものは、この逆方向の計算も実行できます。このようなことから、容易な片方向の計算を公開鍵による処理、逆方向に計算するための知識を秘密鍵と考えることができます。

3.3.1 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すものとなります。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。

公開鍵配布系の代表的なアルゴリズムは、Diffie-Hellman 方式です。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。

3.3.2 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を利用します。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。次に、送信者は、公開されている暗号化鍵を使って平文を暗号化し、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、内容を参照することができます。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、すでに示した対称アルゴリズムでの次の 2 つの問題点を解決することができます。

- 鍵の安全な共有方法が必要
- 通信する相手ごとに異なる鍵が必要

まず、1 番目の問題点は、暗号化鍵を公開してしまうことで解決されます。また、2 番目の問題点は、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ではなくなります。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。

非対称アルゴリズムの代表的なアルゴリズムとして、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、非対称暗号方式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、N. Koblitz と V. S. Miller という 2 名のカナダ人がそれぞれ 1985 年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となっています。現在も研究段階ですが、この暗号方式は、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注目されています。

3.4 暗号アルゴリズムの安全性

ここでは、暗号アルゴリズムの安全性について、いくつかの点から検討してみます。まず、鍵長による安全性を考えてみます。ここでは、アルゴリズム自体には欠陥がなく、力ずくでしか解くことができないものと仮定します。現在、鍵長が 56 ビットである DES による暗号を解くために 56 時間かかっています。これが 1 秒間で解けるようになったとしても、鍵長を 128 ビットとしたときには約 150 兆年を費やさなければなりません。このため、鍵長としては 128 ビット程度あれば十分であると考えられるでしょう。

たとえば、1995 年に得られた推測値では、投入資金として 100 億ドルを費やせば DES は 1 秒間で解けると試算されています。

非対称暗号では、鍵長自体が大きなものとなるため、鍵空間自体を力ずくで解くことは現実的なものとはなりません。ただし、非対称暗号では、素因数分解や離散対数計算などを利用した数学的処理による攻撃の近道があります。このため、非対称暗号で対称暗号と同等の強度を達成するためには、より長い鍵長が必要となります。たとえば、56 ビットの対称暗号と同等の強度を非対称暗号（RSA）で達成するためには 384 ビットが必要であり、128 ビットの対称暗号と同等の強度を達成するためには 2304 ビットが必要となります。

3.5 電子署名

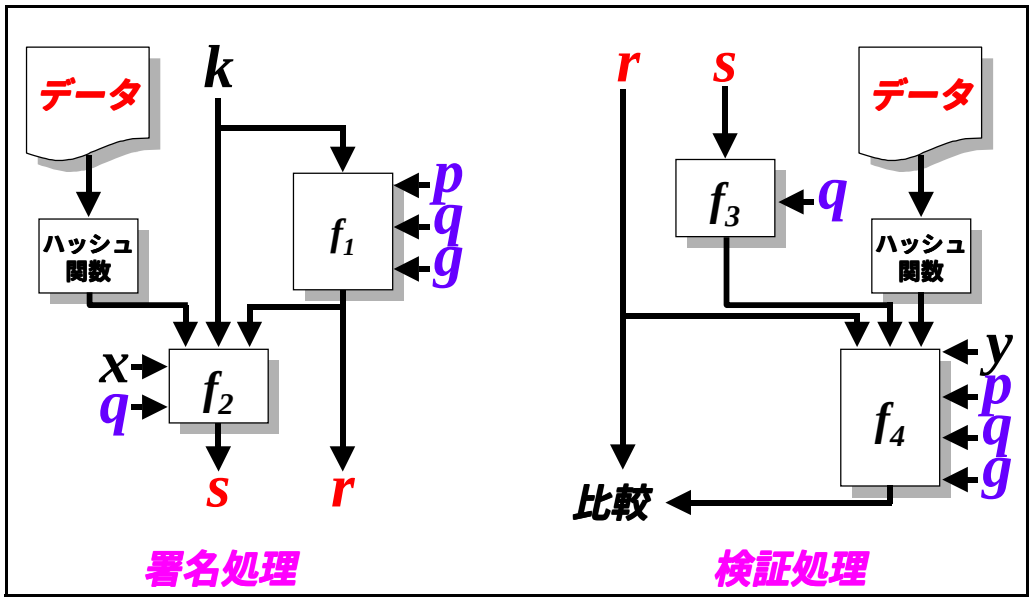
電子署名は、実世界での署名や印鑑押捺などと同様に、内容が改竄されていないことを保証するために作成者がデータに付加する「印」で、デジタル署名とも呼ばれています。電子署名は、通常メッセージダイジェストと非対称暗号技術によって実現されています。

このうち、メッセージダイジェストは、任意のデータからそのデータ特有と見なせる百数十ビット程度の短い情報を抽出する技術です。このときには、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。そして、メッセージダイジェストは、元データの「指紋」として扱うことができます。

メッセージダイジェストとして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を抽出します。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を抽出します。

このようなメッセージダイジェストによる一致証明と、暗号化による署名データの一意性によって、データが改竄されていないことを検証できます。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たものを署名データとします。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、電子署名には、次に示すような DSA（Digital Signature Algorithm）風アプローチという方法があります。



ここで、k は乱数、p、q、g は公開データ、x は署名者の秘密鍵、y は署名者の公開鍵をそれぞれ表しています。この図で示すように DSA 風アプローチでは、署名処理で生成された s と r を使って検証処理が実施されます。ここでもっとも興味深いのは、受け渡される r と元データに何ら関係がないにもかかわらず、r によって元データの完全性を検証できるという点にあります。

4　電子認証技術

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もので、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るため、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 によるガイドラインでは、次のように認証手段を分類しています。

- 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 番目や 2 番目の方法は、入退室管理などの用途では利用できますが、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 種類のものがあります。

- パスワード認証
- チャレンジ & レスポンス
- ワンタイムパスワード
- Kerberos
- 公開鍵証明書

4.1 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。

このように、パスワード認証では、パスワード自体や暗号化されたパスワードが直接ネットワーク上で受け渡されます。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、インターネット環境でのパスワードの受け渡しでは、Base64 による符号化が併用されてい

る例もあります。このようなことから、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。

4.2 チャレンジ & レスポンス

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、ユーザの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。

4.3 ワンタイムパスワード

ワンタイムパスワードには、ソフトウェア方式とハードウェア方式があります。ソフトウェア方式によるワンタイムパスワードでは、あらかじめユーザ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、その内容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た後、次回の認証に備えてその値を保存します。

4.4 Kerberos

Kerberos では、あらかじめすべての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 では、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されたセッション鍵（＝チケット）と、ユーザ A が利用するためのセッション鍵データを、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。ここまでの処理の後、ユーザ A とユーザ B は、互いのセッション鍵を使ってデータを暗号化して通信します。

4.5 公開鍵証明書

これまでに示してきた認証技術には、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は、安全性は高いものとなっていますが、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、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。

公開鍵証明書方式では、公開鍵証明書と秘密鍵による電子署名処理によって認証が実施されます。また、ユーザが署名したデータを公開鍵証明書中の公開鍵で復号することで、ユーザの身元を確認することができます。公開鍵証明書方式では、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。

公開鍵証明書方式では、非対称暗号技術を利用することで認証処理後の通常の通信内容を保護することもできます。また、電子署名によってユーザを特定できるため、否認を防止できます。さらに、ユーザの素性が認証局（CA: Certificate Authority）による保証を経たデータとして証明書中に記載されているため、改めてユーザからの入力を要求せずにその情報を利用することができます。このように、公開鍵証明書方式は、現時点でもっとも優れた認証方式となっています。

5　公開鍵証明書と公開鍵基盤

5.1 公開鍵証明書

公開鍵証明書は、非対称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。公開鍵を保証するには、次の 2 つの方法が代表的です。

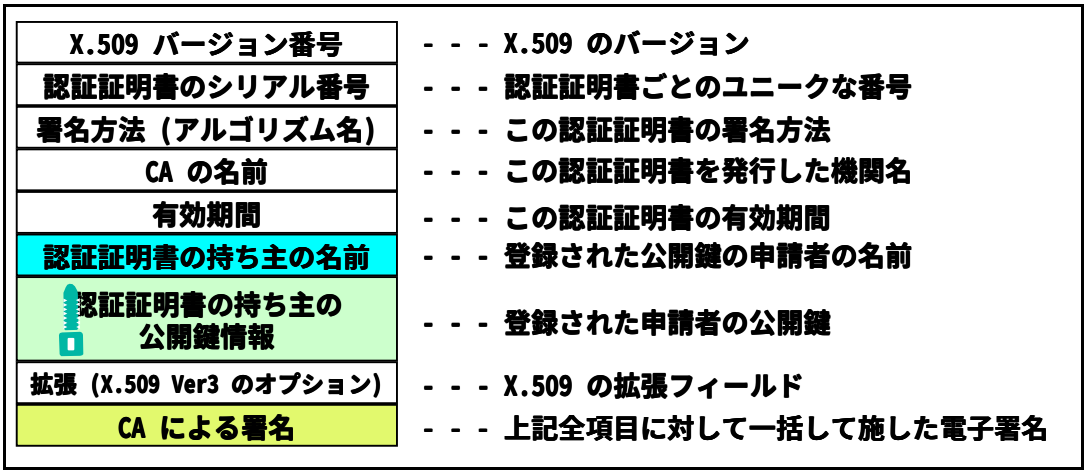
- 草の根的解決（PGP）
- 信頼された第三者機関による保証

草の根的解決は、「友達の友達は友達」方式と言えます。つまり、信頼関係の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することができます。このような PGP の信頼モデルでは、電子署名の署名者への信頼率をパラメータとして公開鍵の持ち主を信頼できるかどうかが決定されます。

このような草の根的解決は、他の仕組みを必要としないため、初期段階での普及が容易なものとなります。ただし、ユーザ層が拡大していくと信頼性が低下してしまいます。また、個人への信頼と、その個人による保証への信頼を区別する必要もあります。

これに対して、信頼された第三者機関による保証では、公開鍵証明書認証局（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るようになります。これは、実社会での印鑑証明書による取引と同様のものと考えることができます。公開鍵証明書とは、認証局が自らの責任の基にユーザと公開鍵との結び付きを証明したものとなります。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とに、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ものが公開鍵証明書の主流となっています。

次の図では、公開鍵証明書の内容を示しています。



このうち特に重要となる部分は、認証証明書の有効期間、持ち主の名前、公開鍵情報となります。このような公開鍵証明書を取得した際に、その正当性を検証するには、最後尾の認証局（CA）による署名を除いた内容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、残りの「CA による署名」を認証局の公開鍵で復号して得られるメッセージダイジェストと照合します。そして、この 2 つのメッセージダイジェストが一致したときには、公開鍵とその持ち主の結び付きが証明されたことになります。

このような証明方法で、認証局の公開鍵はどのように保証されるのでしょうか。認証局の公開鍵自体は、その認証局の上位に位置する認証局の公開鍵証明書によって保証されています。そして、最上位に位置する認証局の公開鍵は、利用するソフトウェアに添付して配布したり、その認証局自体によって公開鍵証明書が発行されることで保証されます。このような階層構造によって、公開鍵証明書の拡張性が確保されています。

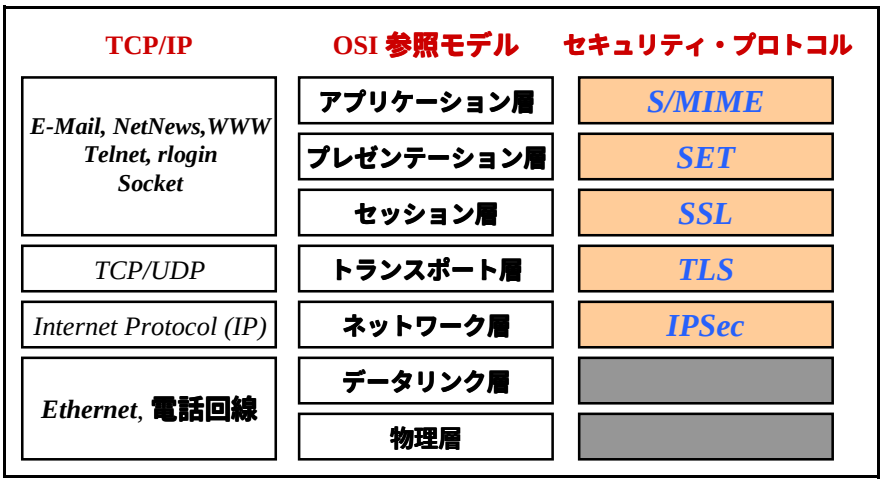
5.2 公開鍵基盤

公開鍵基盤（PKI: Public Key Infrastructure）は、非対称暗号技術を広範囲で利用するための枠組みです。公開鍵基盤では、発行対象、鍵管理、身元確認、運用方針、証明書の有効期限などの規則をあらかじめ定めておく必要があります。また、各認証局の連鎖も、ある証明書の有効性を確認するための証明書パスとして必要となります。このような証明書パスは、次のいずれかの検出方法によって保証されることになります。また、両方の方法を利用できたときには、より効率よく証明書パスを検出できるようになります。

- ある認証局の名前から、その認証局の公開鍵に対して公開鍵証明書を発行した親認証局の証明書を検出する
- ある認証局の名前から、その認証局が他の認証局の公開鍵に対して発行した証明書を検出する

6　セキュリティプロトコル

ここでは、インターネット環境でのセキュリティプロトコルについて説明します。次の図では、ここで説明する各セキュリティプロトコルと、TCP/IP や OSI 参照モデルとの関係を示しています。



6.1 S/MIME

S/MIME（Secure/MIME）は、米国での民間の暗号技術研究 / 開発の総本山的な存在である RSADSI 社によって提唱されました。また、PGP と比較されることが多いですが、S/MIME は、当初から標準規格となるべくして誕生してきたセキュリティプロトコルです。S/MIME は、公開鍵暗号のデファクトスタンダードである PKCS（Public-Key Cryptography Standard）を電子メールに適用するために作成され、1997 年に Netscape Messenger や Microsoft Outlook Express に採用されたことでシェアを一気に拡大しました。

S/MIME は、MIME（Multipurpose Internet Mail Extensions）に準拠し、可視コードの処理などは MIME に依存したものとなっています。MIME 自体は、インターネットメール標準を規定している RFC822 に対する機能拡張であり、7 ビットのフラットなテキストデータしか扱えないインターネットメール標準に対してマルチメディア、7 ビット以外の文字セット、構造を持つデータなどを扱えるようにしています。また、MIME の機能を利用して電子メールにセキュリティ機能を提供しようとする規格として RFC1847 が策定されています。RFC1847 の Security Multiparts for MIME には署名メッセージと暗号化メッセージがありますが、S/MIME では署名メッセージのみが利用されています。そして、S/MIME での暗号化には、PKCS 独自の暗号技術が利用されています。

実際の S/MIME の利用では、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を、受取人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、受取人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、署名付きデータには、次の 2 種類の表現形式が利用できます。

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

SignedData では、元データと署名データを 1 つにまとめて、PKCS 形式に構造化します。ただし、PKCS と互換性のないツールでは読み取ることができなくなってしまいます。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、通常のメーラでも読むことだけはできます。

このような S/MIME では、対称暗号として RC4/40 か DES-EDE3、非対称暗号として RSA か Diffie-Hellman が最低必要条件となっています。また、電子署名では、一方向関数として MD2、MD5、SHA-1、非対称暗号として RSA か DSA が最低必要条件となっています。

6.2 SET

SET（Secure Electronic Transactions）は、クレジットカードを決済手段として利用する電子商取引の規格で、Visa と MasterCard によって 1997 年に提案され規格化されました。実際には、SET は、クレジットカードによる決済をネットワーク経由で安全に実施することを目的としているため、商品選択などの決済以外の処理には関与していません。

SET における暗号化処理では、対称暗号技術と非対称暗号技術がともに利用され、S/MIME と同様に Digital Envelope が使用されています。実際に利用されている暗号化アルゴリズムは、56 ビットの DES と 1024 ビットの RSA という、特に DES の方は、現在のレベルから考えると、それほど強力な暗号技術ではありません。また、SET では、署名用と鍵配送用として、2 種類の公開鍵／秘密鍵ペアが使われます。ただし、クレジットカード所持者は、署名用の鍵ペアのみを用意すれば済むようになっています。

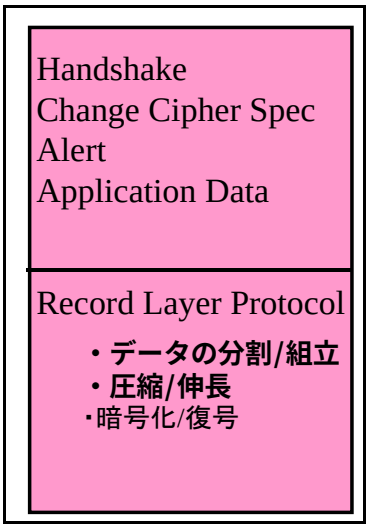
また、SET における暗号化技術では、Digital Envelope の安全性を高めるために OAEP（Optimal Asymmetric Encryption Padding）が利用されています。通常、Digital Envelope の大きさは非対称方式の鍵長程度であり、SET では 1024 ビットとなります。これに対して、Digital Envelop 内に収める対称方式の鍵長が 56 ビットと短いため、Digital Envelope 内には空白部分が発生します。OAEP では、対称鍵の他にカード情報、暗号化されたデータのハッシュ値、乱数などのさまざまなデータを挿入することで、平文を知らずに Digital Envelop 部分を推測できないようし安全性を高めています。

さらに、SET における暗号化処理では、取引の機密性を高めるために二重署名という独自の処理が利用されています。これは、クレジットカード所持者からの、購入店には口座情報を告げず、銀行には商品内容を知られずに、指定した取引内容を処理したいという要求を満たすための解決策となっています。実際には、最初に購入店向けと銀行向けのメッセージをそれぞれ作成し、一方向関数によって各メッセージのハッシュ値を求めます。そして、それぞれ求めたハッシュ値をもう一方のメッセージに付加し、受取人である購入店と銀行の公開鍵を使ってそれぞれの Digital Envelope を作成します。また、二重署名として、それぞれのハッシュ値を合わせたものも作成します。このようなメッセージと二重署名を購入店と銀行にそれぞれ渡すことで、クレジットカード所持者からの要求を満たすことができます。また、それぞれのメッセージに付加されたハッシュ値にメッセージから生成したハッシュ値を合わせ二重署名と照合することで、メッセージを検証することもできます。

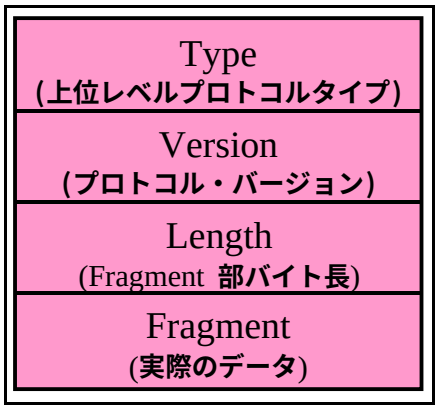
6.3 SSL

SSL（Secure Sockets Layer）は、Netscape Communication 社が提唱した暗号方式で、クライアント / サーバ間での安全なソケット接続を提供し、多数の暗号化方式に対応しています。ただし、現在のところ SSL は、WWW での利用が主なものとなっています。また、これまで日本では、米国の輸出規制によって 40 ビット鍵暗号までしか利用できませんでした。ただし、Fortify for Netscape という制限解除用パッチや SSLeay というフリーなライブラリを利用することで、128 ビット鍵暗号も利用できるようになっています。

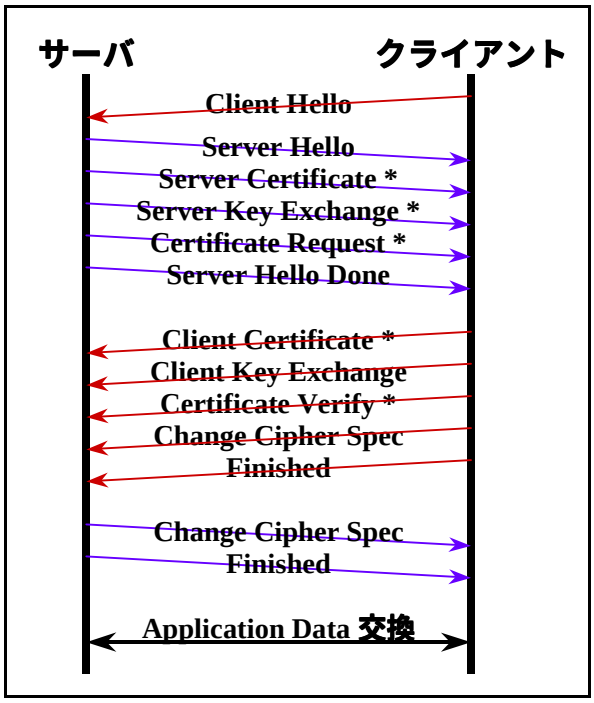
SSL では、OSI 参照モデルのセッション層を次のように 2 つに分けています。



このうち Record Layer Protocol では、2^{14} バイト以下にデータを分割し、必要に応じて圧縮処理、認証データ付加、暗号化処理を実施して、次のようなデータ構造体を生成して送受信に利用しています。



Handshake Protocol では、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するなどの SSL での実質的な要となっています。実際の Handshake Protocol でのやりとりは、次の図のように複雑なものとなっています。



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 か証明書の代わりとなる Server Key Exchange 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を実施するために、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、暗号化と圧縮のアルゴリズムが決定されますが、これらのメッセージには時間データが含まれるために、接続の鮮度が保証されます。最初にクライアントから対応できる暗号化仕様や圧縮方式が示され、それに対して今回利用する暗号化仕様や圧縮方式がサーバから返されます。

また、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となります。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、サーバの公開鍵で暗号化されます。サーバでは、クライアントから受け取ったデータをプリマスタシークレットとし、SHA-1 と MD5 の 2 種類の一方向関数アルゴリズムを使って 48 バイトのマスタシークレットを生成します。

サーバがクライアントの認証を実施する際にクライアントから送られる Certificate Verify も、Client Key Exchange と同様の方法によって作成されます。ただし、Certificate Verify では、マスタシークレットなどに対する MD5 と SHA-1 のハッシュ値がクライアントの秘密鍵で暗号化されています。また、SSL では、暗号化鍵、初期化ベクタ、MAC 計算用秘密データなどとして利用するキーブロックも生成されます。

このように機能している SSL は、対称暗号アルゴリズムとして RC2、RC4、IDEA、DES、3DES、Fortezza、鍵交換アルゴリズムとして RSA、Diffie-Hellman、Fortezza などをサポートしており、一方向関数としては MD5 と SHA-1 を併用しています。

6.4 TLS

TLS（Transport Layer Security）は、IETF で標準化作業中のプロトコルで、本質的な部分は SSL と同様のものとなっています。SSL との違いは、MAC 計算アルゴリズムやマスタシークレットなどの計算アルゴリズムが変更されたことと、Fortezza が非対応となったことです。

6.5 IPSec

IPSec では、IP（Internet Protocol）自体にセキュリティ機能を付与します。ただし、IPSec は、セキュリティ機能を持たない IP と共存でき、次世代 IP である IPv6 だけでなく現行の IPv4 でも利用できます。

また、他のセキュリティプロトコルでは鍵管理方式が含まれていましたが、IPSec では鍵管理方式は分離され、任意の方式を利用することができます。このような鍵管理方式の柔軟性は、Security Association（SA）と Security Parameters Index（SPI）によって実現されています。Security Association（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、Security Parameters Index（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。

現在、このような自動鍵管理方式として標準化作業中であるものに、Diffie-Hellman 方式をベースとした鍵配送プロトコルである Oakley があります。また、SA を構築したり管理するためのメッセージフォーマットを規定するプロトコルとして、ISAKMP（Internet Security Association and Key Management Protocol）も標準化作業中です。

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。このようにトンネルモードではデータ全体が大きくなってしまいますが、このような処理によってオリジナルのヘッダ情報までもが保護されることになります。

IPSec での認証と暗号化は、初期に提示された RFC1825 では認証用データタイプとして IP Authentication Header（AH）、暗号化用データタイプとして IP Encapsulating Security Payload（ESP）がそれぞれ規定されていました。ただし、IPSec の最新モデルでは、ESP に認証機能が含まれてしまっているため、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。

IP Authentication Header（AH）は、IP データグラム全体から認証データを抽出して付加するための新たなヘッダタイプとして制定されました。現在提唱されている認証データの抽出方式には、メッセージダイジェストの抽出アルゴリズムとして MD5 を用いた方式などがあります。また、現在の IP Encapsulating Security Payload（ESP）の規格では、ペイロードデータにブロック長を調整するための乱数データなどが付加された上で、DES CBC 方式によって暗号化されるという方式が定められています。

6.6 PKIX

PKIX（Public Key Infrastructure X.509）は、インターネット全体での PKI 実現のための試みとして、証明書の内容、証明書の管理や運用のためのプロトコル、証明のために必要となるディレクトリ管理プロトコルなどの広範囲におよぶプロトコルや体系を標準化するために IETF によって実施されています。ただし、IETF では SPKI（Simple Public Key Infrastructure）などの他の作業グループも活動しているため、今後の動向には不明確な部分もあります。

次に、現在 PKIX で検討されている項目を示します。

- Certificate and CRL Profile
- Certificate Management Protocols
- Operational Protocols - LDAPv2
- Certificate Policy and Certification Practices Framework
- Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates
- Operational Protocols: FTP and HTTP
- Online Certificate Status Protocol - OCSP
- Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates
- Certificate Request Message Format
- Certificate Management Message Formats
- Certificate Management Messages over CMS
- LDAPv2 Schema
- Caching the Online Certificate Status Protocol
- WEB based Certificate Access Protocol-- WebCAP/1.0 (56291 bytes)
- ENHANCED CRL DISTRIBUTION OPTIONS
- Time Stamp Protocols
- Data Certification Server Protocols
- PKIX Roadmap

このようなさまざまな項目が検討され標準化されていくことで、これまで用途によって個別に利用されてきた技術が、現在構築されつつある IP バックボーン上でのあらゆる認証、秘匿性、安全性、否認防止を実現する PKI バックボーンによって統合されていくのではないかと考えられます。