■3 群（コンピュータネットワーク）－ 7 編（コンピュータネットワークセキュリティ）

6 章 セキュリティプロトコル

（執筆者：寺田真敏）［2009 年 2 月 受領］

■概要■

盗聴，改ざん，なりすましなどの不正な行為からデータを保護する機構として，**表 6･1** に示すような，様々なセキュリティプロトコルがある．

表 6・1 プロトコル階層ごとのセキュリティプロトコル

OSI 階層	プロトコル	内 容
アプリケーション層	XML	汎用データ記述言語である XML には，データを含む任意の情報に対するディジタル署名及び暗号文を XML データとして表現するための標準である XML 署名及び XML 暗号がある．
	S/MIME	電子メールの暗号化と電子署名に関する標準であり，MIME（Multi-Purpose Internet Mail Extensions）をベースとしている．
	PGP	PGP（Pretty Good Privacy）は，Philip Zimmermann 氏によって開発された暗号化のためのツールで，データの暗号化，電子署名などの機能を備えている．
	SET	SET（Secure Electronic Transactions）は，インターネットを通じて安全にクレジットカード決済を行うための仕様として策定された．
	SHTTP	SHTTP（Secure Hyper Text Transfer Protocol）は，HTTP にデータを暗号化して通信する機能を付加したプロトコルであり，サーバとブラウザの間の通信を暗号化する機構を提供する．SSL を HTTP に適用した HTTPS が普及したため，現在ではほとんど使われていない．
	SSH	SSH（Secure Shell）[1]は，rlogin（リモートログイン），rsh（リモートシェル）や rcp（リモートファイルコピー）の認証と暗号通信機構を強化したものである．SSL と同様にアプリケーション層と TCP/IP との間にデータセキュリティ層を提供することもできる．
トランスポート層	SSL/TLS	Web ブラウザとサーバ間のセキュア通信を実現するために Netscape 社により開発されたプロトコルである．Web 上の事実上の標準となっている．TLS は，SSL 3.0 を元に若干の改良を加えた仕様であり，RFC 2246 として IETF で標準化されている．
	Socks	ソケットインタフェースレベルでのトランスポートゲートウェイとして機能する．Socks v5 は，RFC1928 [2]として IETF で標準化されている．
ネットワーク層	IPSec	IP プロトコルのためのセキュリティ機構で，あらゆる通信を IP 層で暗号化（あるいは認証）する機構を提供する．
データリンク層	PPTP	PPTP（Point-to-Point Tunneling Protocol）[3]は，Microsoft，Ascend Communications，3COM が中心となり策定した PPP をトンネリングするためのプロトコルである．認証と暗号化については PPP の機能を利用している．

	L2F	L2F（Layer 2 Forwarding）[4]は，Cisco Systems が Cisco IOS 上で実装したトンネリングプロトコルである．認証と暗号化については PPP の機能を利用している．
	L2TP	L2TP（Layer2 Tunneling Protocol）[5]は，PPP をトンネリングするためのプロトコルであり，PPTP と L2F が統合された仕様となっている．

セキュリティプロトコルを用いたセキュリティの確保にあたっては，仕様自身の安全性だけではなく，仕様に基づいた実装についても安全性を考慮する必要がある．本章で取り上げるセキュリティプロトコルにおいても，国内で利用されるソフトウェアなどの製品のぜい弱性対策情報を中心に収集／蓄積するぜい弱性対策情報データベースである JVN [6]やインターネットにおけるシステムのぜい弱性対策並びにインシデント対策を推進している機関 [7] に実装に伴うぜい弱性が適宜報告されていることにも注意を払って欲しい．

Vulnerability Note VU#349113, isakmpd fails to handle ISAKMP packets with “Payload Length” of zero（2004 年）

鍵交換プロトコル ISAKMP（Internet Security Association and Key Management Protocol）のフィールド長の処理に関するぜい弱性である．ISAKMP と IKE（Internet Key Exchange）を実装した OpenBSD isakmpd において，フィールド長がゼロのパケットを処理する際に，長さを適切に処理しないために無限ループ処理に陥ってしまう．

JVNDB-2002-000311, IBM AIX が実装する IPSec の esp4_input( ) 関数におけるカーネルパニックが発生するぜい弱性（2002 年）

BindView RAZOR により報告された IPSec パケットの認証データフィールド長の処理に関するぜい弱性である．IPSec の複数の実装には，小さい IPSec パケットを処理する際に，認証データフィールドの長さを適切に処理しないために整数オーバーフローが発生してしまう．

JVNDB-2008-001416, OpenSSL TLS ハンドシェイクにサービス運用妨害（DoS）のぜい弱性（2008 年）

TLS 実装として広く使われている OpenSSL に，TLS ハンドシェイク処理にかかわるぜい弱性が報告された．Server Key Exchange メッセージを省略した TLS ハンドシェイク処理が行われた場合，サービス不能状態に陥る可能性がある．

JVNDB-2002-000174, OpenSSL の ASN.1 ライブラリにおけるサービス運用妨害（Dos）のぜい弱性（2002 年）

OpenSSL の ASN.1 ライブラリが不正にエンコードされた証明書を適切に処理しないという問題である．このぜい弱性は，SSL/TLS，S/MIME，PKCS#7 並びに証明書作成など OpenSSL を利用するアプリケーションに影響を与えた．

JVNDB-2002-000172, OpenSSL の SSL2/SSL3 ハンドシェイクにおけるバッファオーバーフローのぜい弱性（2002 年）

SSL 実装として広く使われている OpenSSL に，リモートから攻略可能なバッファオーバーフロー問題，並びに ASN.1 ライブラリのエンコーディング上の問題など，計五つのぜい弱性が報告された．これらぜい弱性のうち，SSLv2 ハンドシェイクプロセスでのぜい弱点については，Apache Web サーバと mod_ssl を介して伝播するワームである Apache/mod_ssl（Slapper）が流布する際に悪用された．

CA-2000-05, Netscape Navigator Improperly Validates SSL Sessions（2000 年）

ACROS Security Team により報告された Netscape Navigator の SSL セッション取扱いに関するぜい弱性である．Netscape Navigator は，Web サーバとの間で新たに SSL セッションを確立する際には証明書の諸条件を正しく確認するが，有効な SSL セッションが存在する場合には証明書の諸条件を再確認しないために，Web サイトのなりすましなどに悪用されてしまう可能性がある．

Vulnerability Note VU#328163, Microsoft Windows XMLHTTP component allows remote access to local data sources（2002 年）

HTTP を利用して XML データを送受信するための XMLHTTP ActiveX コントロールにおけるぜい弱性であり，ローカルシステムのデータを Web サイトに送信するために悪用される可能性があった．

【本章の構成】

本章ではプロトコル階層に沿い，より汎用的なセキュリティプロトコルとして，IP 層では IPSec，トランスポート層では SSL/TLS，アプリケーション層では S/MIME，XML を取り上げる．

■参考文献

1) T. Ylonen et.al., “RFC4251: The Secure Shell (SSH) Protocol Architecture,” 2006. http://www.ietf.org/rfc/rfc4251.txt
2) M. Leech et.al., “RFC1928: SOCKS Protocol Version 5,” 1996. http://www.ietf.org/rfc/rfc1928.txt
3) K. Hamzeh et.al., “RFC2637: Point-to-Point Tunneling Protocol (PPTP),” 1999. http://www.ietf.org/rfc/rfc2637.txt
4) A. Valencia, M. Littlewood, and T. Kolar, “RFC2341: Cisco Layer Two Forwarding (Protocol) "L2F",” 1998. http://www.ietf.org/rfc/rfc2341.txt
5) W. Townsley et.al., “RFC2661: Layer Two Tunneling Protocol "L2TP",” 1999. http://www.ietf.org/rfc/rfc2661.txt
6) Japan Vulnerability Notes (JVN), http://jvn.jp/
7) US-CERT Vulnerability Notes, http://www.kb.cert.org/

■3 群 － 7 編 － 6 章

6-1 IPSec

（執筆者：寺田真敏）[2009 年 2 月 受領]

6-1-1 はじめに

IP プロトコル自身は，データの盗聴や改ざんを防ぐセキュリティ機構を備えていない．IPSec（IP Security Protocol）は IP プロトコルのためのセキュリティ機構で，特定のアプリケーションの通信だけを暗号化（あるいは認証）するのではなく，あらゆる通信を IP 層で暗号化（あるいは認証）することにある．IPSec の開発は，1990 年代前半から始まり，1995 年に RFC1825 から RFC1829 の標準化がなされた．1998 年以降 RFC2401 から RFC2412 として，ESP（Encapsulating Security Payload）への認証機構の導入，IKE（Internet Key Exchange）の標準化などの仕様策定が行われてきた．

仕様改訂並びに拡張の活動は，インターネットの標準化団体である IETF（Internet Engineering Task Force）の IPSec WG [1)]において進められ，2005 年に IKEv2 を含む，IP 層でのセキュリティアーキテクチャ，IP パケットの認証や暗号化プロトコル，鍵交換プロトコルの仕様が RFC4301 から RFC4307 として改訂された．

IPSec は，IPv4 と IPv6 のための共通セキュリティアーキテクチャとしても仕様が検討されており，IPv4 ではオプション扱いとなっているが，IPv6 にはその実装が必須となっている．また，IPSec は，単一のプロトコルではなく複数のプロトコル群の総称であり，その全体構成については RFC4301: "Security Architecture for the Internet Protocol" [2)]に規定されている．

IPSec を構成するプロトコルは，大きく二つのグループに分けることができる（**図 6･1**）．

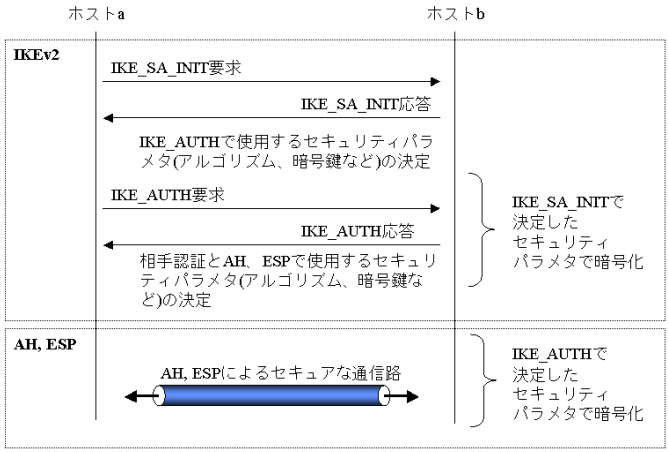

図 6・1　IPSec を用いたセキュアな通信の手順

(1)　IP パケットの認証や暗号化プロトコル

IPSec 通信での認証や暗号化の機能を実現する AH（Authentication Header）及び，ESP（Encapsulating Security Payload）のプロトコルと，そこで使われる認証（HMAC-SHA1 など）

と暗号化（3DES-CBC など）のアルゴリズムを定めている．

(2)　鍵交換プロトコル

IPSec 通信での認証や暗号化のパラメータ（アルゴリズムや暗号鍵など）を決定するための IKE（Internet Key Exchange：鍵交換）プロトコルがある．

6-1-2 Security Association と Security Parameters Index

IPSec の一つの特徴は，暗号や認証方式，鍵交換の仕組みを独立して規定しているところにある．このような独立性を確保するために，IPSec では，一つのセッションに関連するセキュリティパラメータをまとめた SA（Security Association）という考え方を導入している．SA は，認証と暗号アルゴリズム，アルゴリズムで使用する暗号鍵やパラメータなどを格納するフレームワークである．

また，一般に通信を行う場合には複数の通信が同時に行われるので，受信した IPSec パケットがどの SA を使用しているのかを識別する必要がある．このため，宛先 IP アドレス，セキュリティプロトコル(AH, ESP)のほかに, 32 ビットの整数値をもつ SPI(Security Parameters Index)という識別子を IP パケットに格納することで通信に使用する SA との対応付けを行っている（**図 6･2**）．

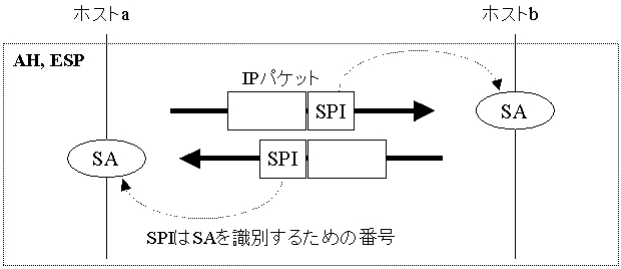

図 6・2　SA と SPI

6-1-3 AH(Authentication Header)

AH [3]は，IP パケットに対する認証データを格納するヘッダで，送信者のなりすまし（認証）と IP パケットの改ざん（完全性）を防ぐための機能を提供する．暗号化通信を使えない場合にも「認証」と「完全性」を保証する通信を提供できる．AH ヘッダには，IP ヘッダ及びデータ部に対して，HMAC-SHA1 や AES-CMAC と呼ぶメッセージ認証子（MAC: Message Authentication Code)生成アルゴリズム [4, 5]を用いて計算した認証データを格納する(**図 6･3**)．ただし，計算対象となる IP ヘッダフィールドは，パケット転送中に値が不変であるか，終点に到達時に値が予測可能であるフィールドとしている．このため，IPv4 の場合には，パケット転送中に値の変化する可能性のある IP ヘッダフィールド，例えば，TTL（Time To Live），TOS（Type of Service），フラグ，フラグメントオフセット，チェックサム，IP オプションについては，認証データの計算にあたりこれらのフィールド値をゼロとする．IPv6 の場合には，クラス，フローラベル，ホップリミットが計算から除外される．

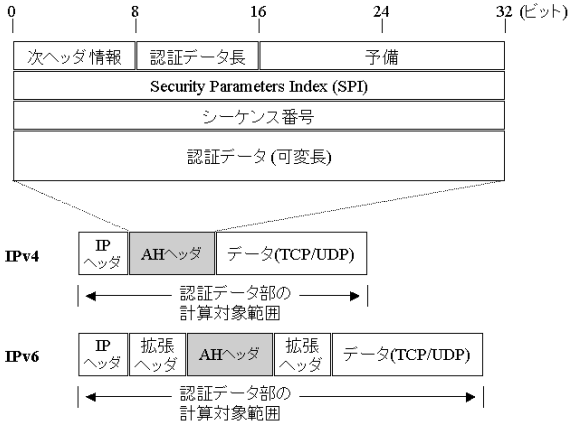

図 6・3　AH ヘッダとトランスポートモードのパケット構成

また，AH には AH ヘッダの前方にオリジナルの IP ヘッダを格納し，AH ヘッダの後方に上位層プロトコル，例えば TCP，UDP，ICMP などを格納するトランスポートモードのほかに，AH ヘッダの前方に新たな IP ヘッダを格納し，AH ヘッダでオリジナルの IP パケット全体を包み込むトンネルモードがある（**図 6･4**）．トランスポートモードはホスト間の通信に利用され，トンネルモードはゲートウェイどうしを介して LAN 間を接続する場合に利用されている．

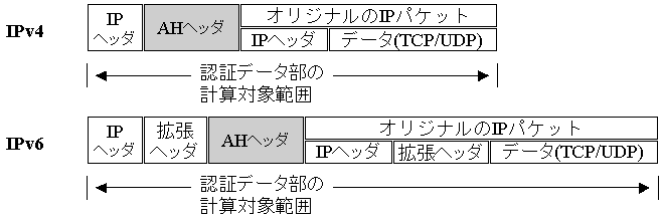

図 6・4　AH におけるトンネルモードのパケット構成

6-1-4　ESP(Encapsulating Security Payload)

ESP[6] は，IP パケットを暗号化し，かつ，その認証データを格納することで，IP パケットの機密性と完全性を保証する（**図 6･5**）．ESP での暗号化は，ESP ヘッダ内の SPI フィールドのインデックス番号に対応した SA で指定されている暗号アルゴリズムやパラメータを使用することになる．暗号アルゴリズムとしては，多くの IPSec 製品が実装している 3DES-CBC や AES などを利用でき，認証アルゴリズムとしては HMAC-SHA1 や AES-CMAC と呼ぶメッセージ認証子を利用することができる [7]．初期の IPSec の仕様では認証のみを行う場合には AH，認証と暗号化を行う場合には AH 並びに ESP を使うこととしていたが，その後の改訂

で ESP にも認証（及びリプレイ防止）機能が組み込まれるようになった．

AH，ESP で利用する暗号アルゴリズム，認証アルゴリズムについては，仕様の経年劣化を踏まえた改訂が行われており，2005 年に発行された RFC4305 [8]では，暗号アルゴリズムについては 3DES-CBC，認証アルゴリズムについては HMAC-SHA1 を必須としている．

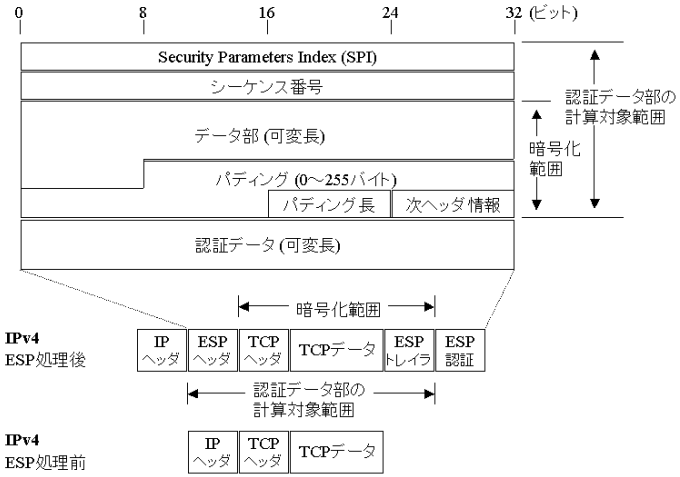

図 6・5　ESP ヘッダとトランスポートモードのパケット構成

ESP には AH と同様にトランスポートとトンネルの二つのモードがある．トランスポートモードは IP パケットのデータ部だけを暗号化し，トンネルモードは IP パケットのヘッダを含めたすべてを暗号化する（**図 6・6**）．通常，トランスポートモードはホスト間の通信に利用され，トンネルモードはゲートウェイどうしを介して LAN 間を接続する場合に利用されている．後者は，ファイアウォール製品の IPSec VPN 機能としてサポートされている（**図 6・7**）．

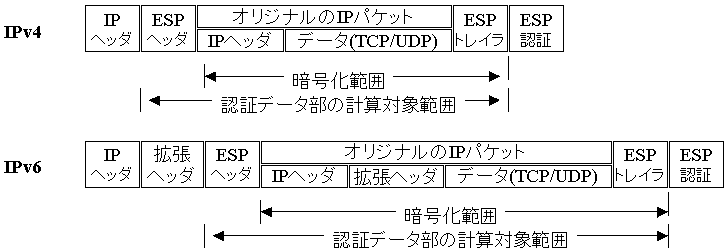

図 6・6　ESP におけるトンネルモードのパケット構成

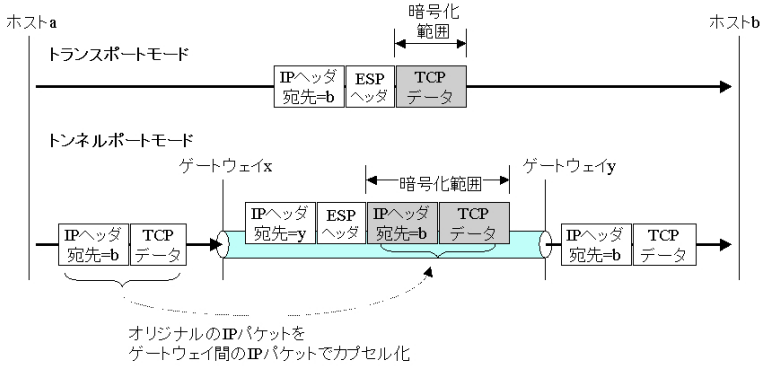

図 6・7　ESP におけるトランスポートモードとトンネルモード

6-1-5 鍵交換プロトコル

AH，ESP による通信に先立って SA を確立するためには，送信側と受信側で認証と暗号化のパラメータを共有する必要がある．IKE v1（Internet Key Exchange）[9] は，汎用的な鍵交換プロトコルのフレームワークを規定した ISAKMP（Internet Security Association & Key Management Protocol）[10] をベースとする IPSec 用鍵交換プロトコルである．この鍵交換プロトコルを用いて通信に必要となる SA を確立する．また，IKE は，AH，ESP とは独立した通信であり鍵交換にあたっては UDP ポート番号 500 を用いた通信を利用している．

IKE v1 を用いた SA 確立は，二つのフェーズから構成される．

- フェーズ 1：互いの正当性の認証を行った後，IKE 自身が安全に情報交換を行うためのセキュリティアソシエーション（IKE SA と呼ばれる）を確立する．正当性の認証のために，IKE では，既知共有鍵（pre-shared key），電子署名（digital signature），公開鍵暗号（public key encryption）の三つの認証方式を用意している．
- フェーズ 2：フェーズ 1 で確立した IKE SA を使い，AH，ESP 通信で使用する認証や暗号化のパラメータの交換を行うことにより，AH，ESP で使用する SA を確立する．

また，2005 年に公開された IKE v2 [11] は，次のような特徴をもつ．

- IKE v2 を用いた SA 確立は，まず IKE_SA_INIT で，二つのノード間の鍵交換を保護するためのセキュリティアソシエーション（IKE SA と呼ばれる）の情報と鍵を生成するためのパラメータを交換する．次に，IKE SA を利用して相互認証と SA の確立を行う．
- 認証方式として，既知共有鍵（pre-shared key），電子署名（digital signature）の二つの認証方式を用意している．
- ISAKMP（RFC 2408），IKE（RFC 2409），DOI（Internet Domain of Interpretation, RFC 2407），NAT（Network Address Translation）トラバーサル，レガシー認証，リモートアドレス取得などを取り込んだ仕様となっている．
- IKE v1 との互換性はないが，IKE ヘッダにバージョン情報を含んでいること，同じ UDP ポート番号を利用するため併用が可能である．

6-1-6 まとめ

IPSec の開発は，1990 年代前半から始まり，1995 年に RFC1825 から RFC1829 として標準化がなされたが，その後の一時期開発活動も低迷していた．しかし，1998 年以降の規格の改訂や大規模エクストラネットである ANX（Automotive Network eXchange）の構築をきっかけに開発や相互接続実験などの活動が活発化した．IPSec WG は 2005 年に終了したが，現在，IP Security Maintenance and Extensions（ipsecme）WG において，IPSec プロトコルの利用に関する活動へと引き継がれている．今後もその動向に注目していく必要がある．

■参考文献

1) IP Security Protocol (ipsec), http://www.ietf.org/html.charters/ipsec-charter.html
2) S. Kent, “RFC4301: Security Architecture for the Internet Protocol,” 2005, http://www.ietf.org/rfc/rfc4301.txt
3) S. Kent, “RFC4302: IP Authentication Header,” 2005, http://www.ietf.org/rfc/rfc4302.txt
4) H. Krawczyk, M. Bellare, and R. Canetti, “RFC2104: HMAC: Keyed-Hashing for Message Authentication,” 2005, http://www.ietf.org/rfc/rfc2104.txt
5) J.H. Song et.al., “RFC4493: The AES-CMAC Algorithm,” 2005, http://www.ietf.org/rfc/rfc4493.txt
6) S. Kent, “RFC4303: IP Encapsulating Security Payload (ESP),” 2005, http://www.ietf.org/rfc/rfc4303.txt
7) C. Madson and R. Glenn, “RFC2404: The Use of HMAC-SHA-1-96 within ESP and AH,” 1998, http://www.ietf.org/rfc/rfc2404.txt
8) D. Eastlake, “RFC4305: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH),” 2005, http://www.ietf.org/rfc/rfc4305.txt
9) D. Harkins and D. Carrel, “RFC2409: The Internet Key Exchange (IKE),” 1998, http://www.ietf.org/rfc/rfc2409.txt
10) D. Maughan et.al, “RFC2408: Internet Security Association and Key Management Protocol (ISAKMP),” 1998, http://www.ietf.org/rfc/rfc2408.txt
11) C. Kaufman, “RFC4306: Internet Key Exchange (IKEv2) Protocol,” 2005, http://www.ietf.org/rfc/rfc4306.txt

■3 群 － 7 編 － 6 章

6-2 SSL/TLS

（執筆者：寺田真敏）［2009 年 2 月 受領］

6-2-1 はじめに

SSL（Secure Socket Layer）プロトコル [1]は Netscape Communications Corporation が提唱したセキュリティプロトコルであり，アプリケーション層（HTTP, TELNET, FTP など）と TCP/IP との間にデータセキュリティ層を提供する．SSL プロトコルは当初，Web ブラウザとサーバ間の暗号通信技術として利用されていたが，現在は，電子メール，LDAP（Lightweight Directory Access Protocol），FTP，IRC（Internet Relay Chat）など様々なアプリケーションに組み込まれている．また，その仕様策定に当たっては，SSL 3.0 まで同社で開発されたが，その後検討の場を IETF に移し 1999 年 TLS（Transport Layer Security）1.0 として標準化された [2]．TLS 1.0 と SSL 3.0 との間に互換性はないが仕様上の差異はごくわずかとなっている．以下，TLS 1.0 について説明するが，これらの説明は SSL 3.0 にも当てはまる．

TLS では，TCP のようなストリーム通信において次の機能を提供する．

- 機密性（Confidentiality）：DES，RC4 などの共有鍵暗号を用いて通信データの暗号化を行う．完全性（Integrity）：SHA，MD5 などのハッシュ関数を用いることにより，MAC（Message Authentication Code）を計算し，このデータを用いて改ざんを検出する．
- 相手認証（Authentication）：X.509 証明書を用いたサーバ認証とクライアント認証を行うことができる．ただし，サーバ認証は必須であるが，クライアント認証はオプショナルである．

そして，このような機能を提供する TLS プロトコルは，大きく二つのグループに分けることができる（**図 6･8**）．

<table>
<tr><td colspan="3">アプリケーション</td></tr>
<tr><td rowspan="2">SSL
/TLS</td><td>ハンドシェイク層</td><td>Handshake Protocol
Alert Protocol
Change Cipher Spec Protocol
Application Data Protocol</td></tr>
<tr><td>レコード層</td><td>Record Protocol</td></tr>
<tr><td colspan="3">TCP/IP</td></tr>
</table>

図 6･8 SSL/TLS のプロトコル構成

(1) レコード層

レコード層で使用されるプロトコルは Record Protocol と呼ばれ，データの機密性とデータの完全性を提供する．TLS では交換するすべてのデータを Record Protocol を用いて転送しており，アプリケーション層から渡されたデータを圧縮／暗号化した後送信する，逆に受信したデータを解凍／復号した後アプリケーション層に引き渡すといった機能を提供する．

(2) ハンドシェイク層

ハンドシェイク層には，Handshake Protocol，Alert Protocol，Change Cipher Spec Protocol，Application Data Protocol の四つのプロトコルが含まれており，TLS を用いた暗号通信に先立ち，プロトコルのバージョンと暗号アルゴリズムのネゴシエーション，サーバ認証とクライアント認証を提供する．認証に当たっては，所有者の公開鍵，所有者名，証明書発行機関名，有効期間が格納された X.509 証明書を用いている．

6-2-2 Record Protocol

Record Protocol は，レコード層におけるデータの圧縮並びに暗号化処理を担うとともに，対抗する通信相手とのデータ交換を仲介する．この Record Protocol では，**図 6･9** に示すフォーマット（構造体）を用いて，圧縮並びに暗号化処理，対抗する通信相手とのデータ交換を実現している．

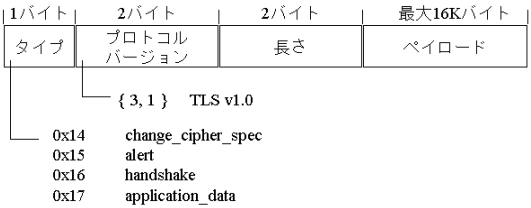

図 6・9　LS レコード層で使用するフォーマット（構造体）

レコード層におけるデータの圧縮並びに暗号化手順を**図 6･10** に示す．まずハンドシェイク層から受信した平文データ（TLSPlaintext）の圧縮処理を行い，圧縮データ（TLSCompressed）を生成する．次に，その圧縮データを対象として MAC（Message Authentication Code）を生成した後，圧縮データと MAC に暗号化処理を施すことにより，最終的に圧縮／暗号化されたデータ（TLSCiphertext）を生成する．ここで，TLSPlaintext，TLSCompressed，TLSCiphertext は図 6･9 に示すフォーマット（構造体）を使用しており，対抗する通信相手には，その時点で合意されている圧縮アルゴリズム，暗号アルゴリズム並びに暗号鍵により処理した TLSCiphertext が転送される．

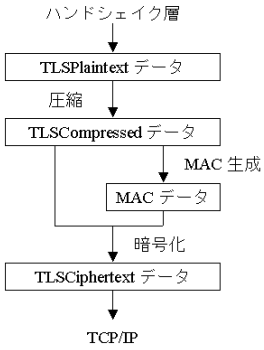

図 6・10　TLS レコード層における圧縮並びに暗号化手順

6-2-3 Handshake Protocol

TLS では，暗号通信に先立ち Handshake Protocol を用いて，暗号アルゴリズム，暗号鍵など暗号通信を開始するために必要となるパラメータをネゴシエーションし，更に，X.509 証明書を用いた認証によりセッションを確立する．**図 6･11** に新たなセッションを確立する際の手順を示す．

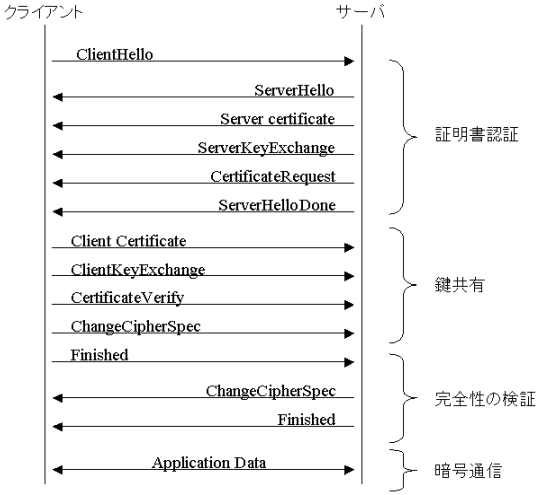

図 6・10　TLS において新たなセッションを確立する際の手順

セッションは，クライアントが利用可能な暗号／圧縮アルゴリズムのリスト，生成した乱数（rnd1）を格納した Client Hello メッセージを送信することで開始される．Client Hello メッセージを受信したサーバは，提示された暗号／圧縮アルゴリズムのリストの中から，利用可能なアルゴリズムを選択する．決定した暗号／圧縮アルゴリズムは，サーバ側で生成した乱数（rnd2），セッション ID と共に Server Hello メッセージとしてクライアントに返送される．その後，サーバは，次のメッセージをクライアントに送信する．

- 決定した暗号／圧縮アルゴリズム，サーバ側で生成した乱数（rnd2），セッション ID を送信する（ServerHello メッセージ）．
- サーバの X.509 証明書を送信する（Server Certificate メッセージ）．
- 証明書が利用できない場合に使用し，RSA，Diffie-Hellman の公開鍵情報を送信する（Server Key Exchange メッセージ）．
- クライアント認証が必要な場合に送信する（Certificate Request メッセージ）．
- 一連のメッセージ送信処理の終了を宣言する（Server Hello Done メッセージ）．

一連のメッセージを受信したクライアントは，サーバの X.509 証明書の確認を行う．ここで，クライアント認証が要求されている場合には，Client Certificate メッセージにクライアントの X.509 証明書を格納して返送する．次に，48 バイトの乱数（PMS: Pre Master Secret）を

生成した後，サーバの公開鍵でこの乱数（PMS）を暗号化し，Client Key Exchange メッセージとして返送することになる．クライアント認証が要求されている場合には，これまでのメッセージのダイジェストを，クライアントの秘密鍵で署名した情報を Certificate Verify メッセージとして送信する処理を行う．

この段階でクライアントは暗号通信に必要となるハンドシェイク処理は終了したので，乱数 rnd1, rnd2, PMS から 48 バイトの秘密鍵 Master Secret を生成し，更に，秘密鍵 Master Secret を用いて送受信鍵となる Server Write Key と Client Write Key，送受信 MAC 鍵となる Client Write MAC Secret と Server Write MAC Secret，そしてブロック暗号の際に使用する送受信用の初期化ベクトル（IV）の六つのパラメータを生成する．これで暗号通信の準備が整ったことになる．クライアントは Change Cipher Spec メッセージを送信し，今後はハンドシェイクで合意した新しいパラメータによる暗号通信を始めることを通知する．

一方，サーバでは，受信した Client Key Exchange メッセージから PRM を取り出し，48 バイトの秘密鍵 Master Secret を生成し，更に，秘密鍵 Master Secret を用いて Server Write Key, Client Write Key などの六つのパラメータを生成する．これでサーバも暗号通信の準備が整うので，クライアントと同様 Change Cipher Spec メッセージを送信し，今後はハンドシェイクで合意した新しいパラメータによる暗号通信を始めることを通知する．

6-2-4 Change Cipher Spec Protocol

Change Cipher Spec Protocol は，Handshake Protocol で決定したパラメータによる暗号通信の利用開始を相手に通知する．通知は Change Cipher Spec メッセージとしてクライアントとサーバのいずれからも送信され，このメッセージを受信するとハンドシェイクで合意した新しいパラメータによる暗号通信を行う状態に移行する．

6-2-5 Application Data Protocol

Application Data Protocol は，Record Protocol を使って，現在有効な暗号パラメータに従いアプリケーションデータの転送を行う．Application Data Protocol 自身は処理機能を持っておらず，アプリケーションが送信するデータは透過的にレコード層へと引き渡され，対抗する通信相手から受信したデータは，レコード層からアプリケーションへの透過的に引き渡されることになっている．

6-2-6 Alert Protocol

Alert Protocol は，ほかのプロトコルの処理中に発生したイベントの深刻度レベルと共にその説明を通知する役割を担っており，20 種強のアラートタイプを用意している．レベルとしては，警告イベントを通知する Warning と，重要イベントを通知する Fatal という二つのレベルが存在する．Fatal レベルのアラートとしては，「受信したメッセージの MAC が不正である（bad_record_mac）」「復号時に問題が発生した（decryption_failed）」「提示されたセキュリティパラメータとして利用できるものがなくネゴシエーションに失敗した（handshake_failure）」など安全な通信環境を阻害するイベントが発生した場合に使用し，Fatal に該当する場合には，直ちにそのコネクションを終了することなる．

6-2-7 まとめ

Web ブラウザとサーバ間のセキュア通信を実現するために Netscape 社により開発された SSL プロトコルは，X.509 証明書を用いた認証を実現するための認証局や Web ブラウザへの認証局証明書の事前導入など，プロトコルを実用化するための周辺環境の整備が普及の一役を担った．現在は，SSL は TLS として仕様の標準化が行われ，電子メール，LDAP ディレクトリサービスなど様々なアプリケーションに展開されている．

■参考文献

1) A. Freier, P. Karlton and P. Kocher, “The SSL Protocol Version 3.0,” 1996, http://wp.netscape.com/eng/ssl3/draft302.txt
2) T. Dierks and C. Allen, “RFC2246: The TLS Protocol Version 1.0,” 1999, http://www.ietf.org/rfc/rfc2246.txt

■3 群 － 7 編 － 6 章

6-3 S/MIME

（執筆者：菊池浩明）［2008 年 9 月 受領］

■概要■

S/MIME（Secure/Multipurpose Internet Mail Extensions）は電子メールにセキュリティ機能を強化したセキュアプロトコルである．提供されるセキュリティ機能は，メッセージの暗号化とディジタル署名であり，これらにより，利用者認証，メッセージ完全性，否認不可性（nonrepudiation），メッセージ秘匿性（confidentiality）が実現されている．MIME は，参考文献 5, 6, 7, 8, 9)の一連の文書で標準化されている，電子メールでの様々なマルチメディア情報を包括的に取り扱うための拡張プロトコルであり，S/MIME で導入されるディジタル署名などもこれらの一つの MIME メッセージとして取り扱われている．それゆえ，MIME データを転送構文として利用しているアプリケーション，例えば，HTTP などのセキュリティ強化にも S/MIME は広く利用可能である．

S/MIME は，多くのプロジェクトの成果を汲んで構成されている．暗号化や署名の形式は，1993 年に標準化された Privacy Enhanced Mail（PEM）[1, 2, 3, 4]に原型を見ることができる．PEM はその後に標準化された MIME との整合性が考慮されなかったため，限定された利用に終り，代って，1995 年に MIME Object Security Services（MOSS）が定められた [10, 11]．また，PEM の元となった ITU-T 勧告 X.509 に従った公開鍵証明書 [20]は，様々なアプリケーションへの対応が可能なように拡張が行われ，IETF でも Public-Key Infrastructure with X.509 WG（PKIX）によってインターネット向けの標準化が進められ，証明書プロファイルの関連文書 [21]にまとめられている．

IETF による標準化作業の一方，RSA Data Security 社（現 RSA Security 社）は独自に標準化を進めていた Public-Key Cryptography Standards（PKCS）[12, 13, 14]に基づいて MIME 形式への拡張を行った．これが S/MIME となり，IETF の S/MIME Mail Security WG が組織されて，1998 年に S/MIME Version 2[16, 17]の標準化が行われた．Version 2 で未対応であった共通鍵による暗号化や異なるセキュリティポリシーをもつドメイン間での通信の問題を対処するため，1999 年に Version 3 が，更に 2004 年には Version 3.1 が RFC [22, 23] になっている．S/MIME の構成要素となっている PKCS #7[13]も，より一般的に（CMS）[18]に標準化されている．現在は，Version 2 を実装したメールクライアントと Version 3 が混在している．

6-3-1 暗号化フォーマット

S/MIME ではメッセージの暗号化とディジタル署名を独立に行うことを許しており，メッセージの暗号化のみを行う“enveloped-only data”，ディジタル署名のみを運ぶ“signed-only data”の 2 種類の形式を定めている．暗号化と署名の両方を行う“signed and enveloped data”は，これらの形式が MIME で表現されていることを利用し，それぞれの入れ子で実現されている．暗号化と署名の順序は，実装に依存してよいことになっているが，多くは署名に対して暗号化を行っている．

暗号化処理を，具体例を用いて説明する．**図 6･11** は enveloped-only data で暗号化したメールの例である．空行を挟んで，MIME ヘッダーと MIME ボディの二つから構成されている．

```
MIME-Version: 1.0
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data;name="smime.p7m"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
   filename="smime.p7m"
From: "Allice" <allice@smime.com>
To: "Bob" <bob@smime.com>
Subject: Hello World

MIAGCSqGSIb3DQEHA6CAMIACAQAxggF9MIIBeQIBADCB4TCBz
IEluYy4xHzAdBgNVBAsTFlZlcmlTaWduIFRydXN0IE5ldHdvc
kfypE5BU088y95m0MghAvEqOVqMFmqbOjlN9mixvKJiKAq5vk
REBfVMJCbx2kCY1Sk1TuDvUBaPXcuuPLx6gAAAAAAAAAAAA
```

図 6・11　Enveloped-only Data の例

`Content-Type: application/pkcs7-mime; smime-type=enveloped-data;` は，MIME ボディが PKCS #7 の Enveloped-Data 型を運んでいることを示す MIME ヘッダである（試験的なヘッダであることを示す x-pkcs7-mime を採用しているクライアントもある）．続いて，転送符号化が 6 ビット符号化 `base64` であることを示すヘッダと添付ファイルを指定する `Content-Disposition:`がきている．後者は，S/MIME をサポートしていないクライアントなどのために用意されているもので，**表 6･2** に従ってファイル名が付けられている．`From:`や `To:`などは，RFC822 に従った通常のメールヘッダである．Subject は暗号化されないことに注意してほしい．

表 6・2　主な SMIME の MIME タイプと添付ファイル名

MIME Type	ファイル名
application/pkcs7-mime	smime.p7m
application/pkcs7-signature	smime.p7s
application/pkcs10	smime.p10

MIME ボディは，Base64 符号化されていてフォーマットが分かりにくいが，復号すると ASN.1 BER によって符号化された PKCS #7 のオブジェクトがバイナリで格納されている．メール本文は，MIME ヘッダとボディからなる通常の MIME エントリーとして前処理が行われた後，MIME の仕様に従って主標準形式（canonical form）に書き換えられる．主標準形式は，転送構文やプラットフォームの違いによって MIME ボディが変化して，ディジタル署名が失敗することを防止するために導入されている．例えば，行末の表現が hCRi だけの OS から hLFi だけの OS へメッセージが送られても，署名の計算は両方とも主標準形式である hCRi と hLFi の両方をつけたかたちで行われるので，メッセージ完全性が保証される．

6-3-2 署名フォーマット

ディジタル署名は PKCS の SignedData で表現され，MIME の `application/pkcs7-mime`

型か multipart/signed 型のどちらかで運ばれる．前者が平文のメッセージを PKCS の中に格納するのに対して，後者は単独の text/plain 型の MIME エントリーとして転送するので，S/MIME サービス未対応のクライアントでもメール本文を読むことができる．このとき，本来は PKCS SignedData の中に平文用に予約された ContentInfor フィールドは空になる．**図 6・12** は，この multipart/signed によって符号化されたメールの例である．

```
Content-Type: multipart/signed;
    protocol="application/pkcs7-signature";
    micalg=sha1; boundary="------88B"
From: "Allice" <allice@smime.com>
To: "Bob" <bob@smime.com>
Subject: Hello again

This is a cryptographically signed message in MIME format.

------88B
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

 This is a clear-signed message.

------88B
Content-Type: application/pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIIKUQYJKoZIhvcNAQcCoIIKQjCCCj4CAQExCzAJBgUrDgMCGgUAMAsGC
B90wggRzMIID3KADAgECAhAxbVtktLNqGzFhMr/3md8AMA0GCSqGSIb3D
FQYDVQQKEw5WZXJpU2lnbiwgSW5jLjEfMB0GA1UECxMWVmVyaVNpZ24gV
azFGMEQGA1UECxM9d3d3LnZlcmlzaWduLmNvbS9yZXBvc2l0b3J5L1JQQ
IFJlZi4sTElBQi5MVEQoYyk5ODFIMEYGA1UEAxM/VmVyaVNpZ24gQ2xhc

------88B--
```

図 6・12　Signed-Only Data の例

ここで，text/plain 型と application/pkcs7-signature 型の二つの MIME エントリーからなっていることが観察できる．したがって，先にテキストがきて，後から署名アルゴリズムの指定が行われることとなる．クライアントが 1 パスで署名の検証ができるように，それらより先にパラメータ micalg でハッシュアルゴリズムを指定することを義務つけている．この例では，SHA1 が用いられている．

6-3-3 PKCS #7 による暗号化・署名処理

暗号文もディジタル署名も，それぞれ，pkcs7-mime と pkcs7-signature という PKCS #7[13]で指定されるオブジェクトである．ここでは，PKCS #7 で定められている処理の概要を述べる．

PKCS は，RSA Data Security 社が定めた暗号化処理に関する標準化文書である．S/MIME に

ついては，#1 RSA 暗号化，#7 暗号文文法，#10 公開鍵証明書要求が関係している．PKCS では，データ形式を抽象化構文 ASN.1（Abstract Syntax Notation One）[24]を用いて定義し，転送構文 BER（Basic Encodeing Rule）[25]に従って符号化することを定めている．アルゴリズムによってパラメータの数や形式が大きく変わることが多い暗号化処理にとって，あいまいさなく柔軟にデータを構造化する ASN.1 は向いている．

PKCS #6 には，いくつかの Content Type が用意されているが，S/MIME ではディジタル署名を運ぶ SignedData と暗号文を運ぶ envelopedData，及び，そのほかの情報を運ぶ data の 3 種類を利用する．これらは，すべて**図 6･13** で示されるデータ構造で示される．ContentType で与えられるオブジェクト識別子がこれら 3 種類の中から一つを指定し，content は指定された型に依存する．

```
ContentInfo ::= SEQUENCE {
  contentType ContentType,
  content [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL}

ContentType ::= OBJECT IDENTIFIER
```

図 6・13　PKCS #7 **定義**(a) General Syntax

ディジタル署名は，**図 6･14** で定められる SignedData の形式をしている．このデータは，メール本文をハッシュするアルゴリズムを指定する digestAlgorithms，発信者（署名者）の X.509 形式の公開鍵証明書を格納する certificates（証明書検証パスの必要な場合は上位の複数の証明書を含む），証明書が廃止されていないことを証明する Certificate Revocation List（CRL）を格納する crls，そして，署名データを格納する signerInfos からなる．PKCS では平文を格納するために用意されている contentInfo は，S/MIME の場合は空にする．CRL は一般に数百キロバイトの大きさがあり，省略されることが多い．

```
SignedData ::= SEQUENCE {
  version Version,
  digestAlgorithms DigestAlgorithmIdentifiers,
  contentInfo ContentInfo,
  certificates [0] IMPLICIT CertificateSet OPTIONAL,
  crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
  signerInfos SignerInfos }

SignerInfos ::= SET OF SignerInfo
```

図 6・14　PKCS #7 **定義**(b) SignedData

暗号文は，図 6･15 のデータ構造で処理されている．EnvelopedData は，各受信者について計算された暗号鍵の情報を格納する recipientInfos，暗号文を格納する encryptedContentInfo からなる．メール本文を暗号化するデータ暗号鍵はランダムにつくられ，受信者の公開鍵で PKCS #1 に従って暗号化される．受信者が複数いるときは，同じデータ暗号鍵を（パディングの情報を変えて）それぞれの公開鍵で暗号化する．暗号文は，ブロック暗号の大きさに応じてパディングされた後，指定されたアルゴリズムで処理される．

```
EnvelopedData ::= SEQUENCE {
  version Version,
  recipientInfos RecipientInfos,
  encryptedContentInfo EncryptedContentInfo}

RecipientInfos ::= SET OF RecipientInfo

EncryptedContentInfo ::= SEQUENCE {
  contentType ContentType,
  contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
  encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }
```

図 6・15　PKCS #7 定義(c) EnvelopedData

S/MIME の実装におけるハッシュ関数や署名などの各種暗号アルゴリズムの必須条件は，バージョンに依存する最も大きな点である．Version 3 におけるこの違いを**表 6･3** で整理しよう．RFC 固有の表現である“MUST（必須）”，“SHOULD（推奨）”を用いている．例えば，MD5 は解析技術の発達により既にその安全性は保証されていないが，Version 2 との下位互換性を保つために受信はできることが要請されている．

表 6・3　各種アルゴリズムと実装における条件

アルゴリズム		受信	送信
ハッシュ関数	SHA1	MUST	MUST
	MD5	SHOULD	–
署名	DSA	MUST	*1
	RSA	MUST	*1
公開鍵暗号	RSA	MUST	MUST
	DH	SHOULD	SHOULD
共通鍵暗号	DES EDE3 CBC	MUST	MUST
	RC2/40	SHOULD	–
	AES	SHOULD	SHOULD

*1: DSA か RSA の少なくともどちらか一つを必須

6-3-4 S/MIME の課題

現在，S/MIME は多くのメールクライアントで実装されており，最も手軽にディジタル署名を利用できるアプリケーションとなってきた．ディジタル署名を取り巻く法整備も充実してきており，今後の電子政府の確立と共に，従来の物理的な印鑑に代わってディジタル署名の利用を推し進める大きなツールとなることが予想される．その一方，次にあげる課題が残されている．

- S/MIME は公開鍵基盤 PKI の存在に大きく依存している．PKI で用いられる証明書は，SSL 用のドメインを対象としたものが主流であり，個人用の S/MIME 証明書は普及に至っていない．
- 一つの言語に対して複数の文字コードが混在しているため，メール本文を一意に表現する方法が確立していない．それゆえ，pkcs7-signature のように平文を送る形式の場合に署名検証に支障が出る可能性がある．

- 有効期限前に廃止された無効な公開鍵証明書が利用される恐れがある．これを完全に防止するためには，CRL を頻繁に交換したり，証明書状態サーバを設けてオンラインで検証したりするコストが生じ得る．

■参考文献

1) Linn, J., “Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures,” RFC 1421, DEC, Feb. 1993.
2) Kent, S., “Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management,” RFC 1422, BBN, Feb. 1993.
3) Balenson, D., “Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, andIdentifiers,” RFC 1423, TIS, Feb. 1993.
4) Balaski, B., “Privacy Enhancement for Internet Electronic Mail: Part IV: Notary, Co-Issuer, CRLStoringand CRL-Retrieving Services,” RFC 1424, RSA Laboratories, Feb. 1993.
5) Freed, N., and N. Borenstein, “MIME Part 1: Format of Internet Message Bodies,” RFC 2045, Nov. 1996.
6) Freed, N., and N. Borenstein, “MIME Part 2: Media Types,” RFC 2046, Nov. 1996.
7) Moore, K., “MIME Part 3: Message Header Extensions for Non-ASCII Text,” RFC 2047, Nov.1996.
8) Freed, N., Klensin, J., and J. Postel, “MIME Part 4: Registration Procedures,” RFC 2048, Nov. 1996.
9) Freed, N., and N. Borenstein, “MIME Part 5: Conformance Criteria and Examples,” RFC 2049, Nov. 1996.
10) Galvin, J., Murphy, S., Crocker, S., and N. Freed, “Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted,” RFC 1847, Oct. 1995.
11) S. Crocker, N. Freed, J. Galvin, and S. Murphy, “MIME Object Security Services,” RFC 1848, Oct. 1995.
12) Kaliski, B., “PKCS #1: RSA Encryption Version 1.5,” RFC 2313, Mar. 1998.
13) Kaliski, B., “PKCS #7: Cryptographic Message Syntax Version 1.5,” RFC 2315, Mar. 1998.
14) Kaliski, B., “PKCS #10: Certification Request Syntax Version 1.5,” RFC 2314, Mar. 1998.
15) Chokhani, S. and W. Ford, “Internet X.509 Public Key Infrastructure, Certificate Policy and Certification Practices Framework,” RFC 2527, Mar. 1999.
16) Dusse, S., Homan, P., Ramsdell, B., Lundblade, L. and L. Repka, “S/MIME Version 2 Message Specification,” RFC 2311, Mar. 1998.
17) Dusse, S., Homan, P. and B. Ramsdell, “S/MIME Version 2 Certificate Handling,” RFC 2312, Mar. 1998.
18) Housley, R., “Cryptographic Message Syntax,” RFC 2630, Jul. 2004.
19) Rescorla, E., “Die-Hellman Key Agreement Method,” RFC 2631, Jun. 1999.
20) ITU-T Recommendation X.509 (1997).ISO/IEC 9594-8:1997, Information technology - Open Systems Interconnection - The Directory: Authentication framework.
21) Housley, R., Polk, W., Ford, W. and D. Solo, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 3280 (旧 2459), Apr. 2002.
22) Ramsdell, B., “Secure/Multipurpose Internet Mail (S/MIME) Version 3.1 Certificate Handling,” RFC 3850 (旧 2632), Jul. 2004.
23) Ramsdell, B., “Secure/Multipurpose Internet Mail (S/MIME) Version 3.1 Message Specification,” RFC 3851 (旧 2633), Jul. 2004.
24) X.208-88 CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1), 1988.
25) X.209-88 CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), 1988.

3 群 － 7 編 － 6 章

6-4 XML 署名と XML 暗号

（執筆者：羽田知史）［2009 年 2 月 受領］

6-4-1 はじめに

XML（Extensible Markup Language，拡張可能マーク付け言語）[1]は，1998 年 2 月に，W3C（World Wide Web Consortium）において標準化された汎用データ記述言語である．もともとは，構造化文書の記述言語である SGML（Standard Generalized Markup Language）のサブセットとして標準化が開始されたが，現在では，インターネット上で企業間がデータ交換を行うための汎用データ記述言語としても注目されている．本節では，XML データを含む任意の情報に対するディジタル署名及び暗号文を XML データとして表現するための標準である XML 署名 [2]及び XML 暗号 [3]について解説する．それぞれ，2002 年 2 月及び 2002 年 12 月に，W3C の標準（W3C Recommendation）として，既に公開され標準化作業は終了している．その後，XML 署名に関しては，第 2 版が 2008 年 6 月に公開されている [9]．本稿は，2002 年に公開された XML 署名 [2]に基づいて説明する．

２者間のデータ交換において，ネットワーク上のデータを盗聴，改ざん，なりすましなどから守るためには，本章の前節までに解説されている IPSec や SSL/TLS のような標準プロトコルを用いれば十分である．しかし，これらの標準プロトコルだけでは満たせない要件もある．例えば，企業間のデータ交換で必要となる要件の一つとして，否認不可性（non-repudiation）があげられる．IPSec や SSL/TLS では，交換されるデータに対するディジタル署名は付けられないので，否認不可性は満たされない．しかし，単にディジタル署名が必要なだけなら，前節で紹介された S/MIME を用いることにより解決できる．では，XML 署名や XML 暗号が必要とされる理由とは何だろうか？

そもそも，IPSec，SSL/TLS，S/MIME においては，署名や暗号化の対象となるデータの内部構造については，何も考慮されない．つまり，データ全体に対する署名や暗号化の処理が前提となっている．それに対して，XML 署名及び XML 暗号では，データが XML データの場合に，データ全体に対する処理だけではなく，データの内部構造に依存した処理が可能である．例えば，データの一部分だけを暗号化したり，データのある部分は署名鍵 A で署名し，また別の部分は署名鍵 B で署名する，といったことが可能である．実際，XML データは，構造化されたデータであるので，データの一部を特定することは容易である．したがって，データの内部構造に依存した署名や暗号化を行いたい場合，XML 署名や XML 暗号化の技術が必要となるといえる．

XML 署名及び XML 暗号の両方において，署名及び暗号化の対象となるデータは必ずしも XML データである必要はなく，任意のデータを署名及び暗号化することも可能である．しかし，本節では，上述の観点から，XML データに対する署名及び暗号化に焦点をあてることとする．

6-4-2 XML 署名

XML 署名は，任意のデータに対するディジタル署名を生成及び検証するための処理規則と，生成されるディジタル署名を XML データとしての表現するための構文を定義している．そ

の構文の概要を以下に示す．つまり，生成されるディジタル署名は，<Signature>要素として表現される．

```
<Signature Id?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI? >
      (<Transforms>)?
      <DigestMethod>
      <DigestValue>
    </Reference>)+
  </SignedInfo>
  <SignatureValue>
 (<KeyInfo>)?
 (<Object Id?>)*
</Signature>
```

XML 署名は，3 種類のディジタル署名の形態をサポートしている．<Signature>要素が署名対象のデータに含まれる場合，Enveloped 署名と呼ばれる．また，逆に，<Signature>要素が署名対象のデータを含む場合，Enveloping 署名と呼ばれる．そして，<Signature>要素と署名対象のデータとが別々に存在する場合は，Detached 署名と呼ばれる．

では，この<Signature>要素の構文とその意味の概要を説明する．まず，<Signature>要素は，<SignedInfo>，<SignatureValue>，<KeyInfo>，及び<Object>要素から構成される．<SignedInfo>要素は，署名対象のデータ及び RSA などのディジタル署名アルゴリズムに関する情報が格納される要素であり，XML 署名において最も重要な要素である（詳細は後述）．XML 署名では，RSA などの公開鍵暗号に基づくディジタル署名に加えて，HMAC などの MAC（メッセージ認証コード）も署名アルゴリズムとして利用可能である．XML 署名では，各種アルゴリズムを指定するための識別子が URI として定義されている．<SignatureValue>要素は，<SignedInfo>要素に対して計算されたディジタル署名あるいは MAC の値が格納される．その際，使用される署名アルゴリズムは，<SignedInfo>要素内で指定されたものである．<KeyInfo>要素は，使用された署名鍵に関する情報が格納される．例えば，署名アルゴリズムとして RSA が使用された場合には，署名に使用した秘密鍵に対応する公開鍵証明書が格納される．ただし，<KeyInfo>要素は省略可能である．例えば，検証者があらかじめ署名者の公開鍵証明書を知っていると仮定できる場合は，省略してもかまわない．<Object>要素は，アプリケーションが，タイムスタンプやシーケンス番号などのディジタル署名に関連する任意の情報を格納するための省略可能な要素である．Enveloping 署名の場合は，署名対象のデータは，<Object>要素内に格納される．

前述のように，<SignatureValue>要素に格納されるディジタル署名の値は，<SignedInfo>要素に対して計算されたものである．以下では更に，<SignedInfo>要素の詳細について説明す

る．<SignedInfo>要素は，<CanonicalizationMethod>，<SignatureMethod>，及び 1 個以上の<Reference>要素から構成される．<CanonicalizationMethod>要素では，<SignedInfo>に対するディジタル署名を計算する際に用いる正規化（canonicalization）[4]のアルゴリズムが指定される．同様に，<SignatureMethod>要素では，<SignedInfo>に対するディジタル署名を計算する際に用いるディジタル署名アルゴリズムが指定される．XML 署名では，具体的なアルゴリズムとして，HMAC-SHA1，DSAwithSHA1（DSS），及び RSAwithSHA1 の三つの URI が定義されているが，新たな URI を定義すれば別のアルゴリズムも使用可能である．<Reference>要素は，実際に署名されるデータそのものに関する情報が格納される要素であり，かなり柔軟に設計されている．<SignedInfo>要素には 1 個以上の<Reference>要素を指定できる．つまり，XML 署名では，複数のデータに対する署名を表現できる．署名されるデータは，<Reference>要素の URI 属性において，URI によって指定される．例えば，http://example.com/bar.xml という URI の場合は，example.com という HTTP サーバにある bar.xml という XML データ全体が対象データである．また，bar.xml の一部分に対して署名したい場合は，http://example.com/bar.xml#chapter1 などと指定すればよい．これは，bar.xml 内の chapter1 という ID で指定される要素が署名対象であることを意味する．<Reference>要素には，URI 属性以外に，<Transforms>，<DigestMethod>，及び<DigestValue>要素が格納される．<DigestValue>要素には，URI 属性で指定される署名対象データのハッシュ値が格納される．使用されるハッシュアルゴリズムは<DigestMethod>要素において指定される．XML 署名では，具体的なアルゴリズムとして，SHA1 の URI が定義されている．署名アルゴリズムと同様，新しい URI を定義すれば別のハッシュアルゴリズムも使用可能である．<Transforms>要素は，省略可能であるが，ハッシュ値を計算する前に対象データに対して行う複数の変換処理を指定できる．例えば，対象データの一部分を指定するための変換処理が可能である．XML データの一部分の指定方法として，URI 属性において特定の ID をもつ要素を指定できることを説明したが，XPath フィルタリングと呼ばれる変換処理を用いれば，XPath[5]を用いてもっと複雑な指定が可能である．例えば，XML データ内のある特定の要素名をもつすべての要素を指定したり，あるいは，ある特定の属性がある特定の値をもつようなすべての要素を指定したりできる．そのほか，変換処理の例としては，XSLT 変換[6]や正規化があげられる．

以上，<Signature>要素の概要を説明してきた．以下では，以上の説明をまとめる意味で，<SignatureValue>で指定されるディジタル署名の値が，<SignedInfo>で指定された情報に基づいて，実際にどのように計算されるかを説明する．まず，各<Reference>要素の<DigestValue>要素を計算する必要がある．このために，まず，URI 属性で指定される XML データを取得する．次に，その XML データに対して<Transforms>要素で指定される変換処理を行う．更に，その変換結果に対して<DigestMethod>要素で指定されるアルゴリズムを使用してハッシュ値を計算し，<DigestValue>要素にその計算結果を格納する．この時点で，すべての<Reference>要素の<DigestValue>要素が計算され，<SignedInfo>要素が完成する．その後，<SignedInfo>要素に対して，<CanonicalizationMethod>要素で指定されるアルゴリズムを使用して正規化を行い，その正規化の結果に対して<SignatureMethod>で指定されるアルゴリズムを使用してディジタル署名の値を計算し，<SignatureValue>要素に格納する．

最後に，XML 署名の具体例を紹介する．この例では，署名対象のデータは二つで，それぞれ <Reference> 要素において指定されている．一つ目の <Reference> 要素は，

http://example.com/bar.xml#chapter1 で指定される XML データの一部である．二つ目の <Reference>要素は，#timestamp で指定されている．これは，<Object>要素を指している．前者の場合，署名対象が外部に存在し，後者の場合は，署名対象は<Signature>要素に含まれる．したがって，この XML 署名は，Detached 署名及び Enveloping 署名の一例である．

```
<?xml version='1.0' encoding='UTF-8'?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI=" http://example.com/bar.xml#chapter1">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>xSqlZyXh3LSmjp8PaJG3B2dRUO0=</DigestValue>
    </Reference>
    <Reference URI="#timestamp">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>/QVcMabbPr+eUiaKgSqXsgLpTaQ=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>
    VWF9uiyogGaCDYJ6/wmpdHgKMZqDt23zG5erHB+WAE9iuJCy40Eudg==
  </SignatureValue>
  < Object Id="timestamp">
    <x:Timestamp xmlns:x="http://www.trl.ibm.com/timestamp">2000-01-20T12:00:00 </x:Timestamp>
</Object>
</Signature>
```

前述のように，2008 年 6 月に第 2 版が公開されており [9]，新しい正規化の仕様 Canonical XML Version 1.1[10]が正規化アルゴリズムとして追加されている．

6-4-3 XML 暗号

XML 暗号は，任意のデータを暗号化するための処理規則と，生成される暗号文を XML データとしての表現をするための構文を定義している．その構文の概要を以下に示す．つまり，暗号化した結果の暗号文は，<EncryptedData>要素として表現される．特に，暗号化の対象が XML データで，そのデータ内のある一部分を暗号化した場合は，その一部分は，暗号化結果を表す<EncryptedData>要素によって置き換えられることになる．その際，XML データの中で，暗号化の対象となりうるのは要素あるいは要素の内容のいずれかである．前者の場合，

暗号化されるのは，開始タグから終了タグまでである．つまり，要素名も含めて，要素全体が暗号化される．一方，後者の場合は，開始タグの直後から終了タグの直前までが暗号化される．したがって，要素名に秘匿性を要求する場合は，前者の形態を利用すればよい．

```
<EncryptedData Id? Type? MimeType? Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?
    <AgreementMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue>?
    <CipherReference URI?>?
  </CipherData>
  <EncryptionProperties>?
</EncryptedData>
```

では，<EncryptedData>要素の構文とその意味の概要を説明する．<EncryptedData>要素は四つの属性をもつことができる．Type 属性は，暗号化された XML データが，要素か要素内容かを指定できる．Element が指定された場合は，要素であり，Content が指定された場合は，要素内容である．復号の際には，この値をもとに元のデータを復元できる．MimeType 及び Encoding 属性は，元のデータの MIME タイプと符号化方法を指定する．特に，元のデータが XML データ以外の場合に有効である．例えば，元のデータが BASE64 符号化された PNG ファイルである場合，Encoding 属性は，BASE64 を表す URI が指定され，MimeType 属性は，image/png となる．

<EncryptedData> 要素は，<EncyprtionMethod>，<KeyInfo>，<CipherData>，及び <EncryptionProperties>要素から構成される．<EncyprtionMethod>要素では，暗号化アルゴリズムや暗号化鍵のサイズに関する情報が格納される．XML 暗号では，各種アルゴリズムは URI によって指定される．例えば，トリプル DES，AES，PKCS バージョン 1.5，RSA-OAEP，Diffie-Hellman 鍵共有などの URI が定義されている．<KeyInfo>要素は，暗号化に使用した鍵，あるいは，復号に必要な鍵に関する情報が格納される．<KeyInfo>要素は XML 署名で定義された構文をそのまま再利用しているが，XML 暗号では新たに二つの子要素を定義している．一つは，<EncryptedKey>要素であり，元のデータを AES などの秘密鍵暗号で暗号化した場合に，その暗号化鍵を RSA-OAEP などの公開鍵暗号で暗号化した結果を格納するために使用される．もう一つは，<AgreementMethod>要素であり，Diffie-Hellman 鍵共有などの鍵共有アルゴリズムを用いて，暗号化鍵を共有する場合に，そのパラメータを格納するために使用される．<CipherData>要素には暗号文のデータが格納される．XML 暗号では，2 通りの格納方法を提供している．一つは，暗号文データを BASE64 符号化して，<CipherValue>要素内容として格納する方法であり，もう一つは，外部に暗号文を保管し，<CipherReference>要素の URI

属性にその参照先を格納する方法である．< EncryptionProperties>要素は，アプリケーションが，タイムスタンプなどの暗号化に関連する任意の情報を格納するための省略可能な要素である．

以上，<EncryptedData>要素の概要を説明してきた．以下では，XML データの一部が暗号化される例を示す．以下は，支払い情報を表す XML データの例であり，クレジットカード番号などのプライバシーデータが含まれている．

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>Satoshi Hada </Name>
  <CreditCard Limit='5,000,000' Currency='YEN'>
    <Number>4019 2445 0277 5567</Number>
    <Company>VISTER</Company>
    <ExpirationDate>2010-04-01</ExpirationDate>
  </CreditCard>
</PaymentInfo>
```

ここでは，<CreditCard>要素を暗号化する例を考える．より詳細には，Limit 属性などは暗号化せずに，その内容，つまり，三つの子要素<Number>，<Company>，及び<ExpirationDate>を暗号化したい場合を考える．この場合は，以下のように，<CreditCard>要素そのものはそのままで，その三つの子要素が，一つの<EncryptedData>要素によって置き換えられる．その Type 属性は，Content を表す URI である．暗号化に使用したアルゴリズムは，AES の 128 ビットの初期化ベクトルをもつ CBC モードである．また，暗号化に使用された秘密鍵は，<KeyInfo>要素内の<EncryptedKey>要素に，暗号化されて格納されている．その暗号化には，アルゴリズムとして RSA（PKCS バージョン 1.5）が用いられ，Alice の公開鍵で暗号化されている．

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>Satoshi Hada</Name>
  <CreditCard Limit='5,000,000' Currency='YEN'>
    <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
      Type='http://www.w3.org/2001/04/xmlenc#Content'>
    <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Alice</KeyName>
```

```
            </KeyInfo>
            <CipherData>
              <CipherValue>k3453rvEPO0vKtMup4NbeVu8nk=</CipherValue>
            </CipherData>
          </EncryptedKey>
        </KeyInfo>
          <CipherData>
            <CipherValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</CipherValue>
          </CipherData>
        </EncryptedData>
      </CreditCard>
    </PaymentInfo>
```

6-4-4 まとめ

本節では，XML 署名と XML 暗号の概要を説明した．いずれにおいても，XML データを署名及び暗号化するための汎用的な仕組みを定義している．特に，データ構造に依存した署名及び暗号化が可能となる．しかし，いずれにおいても，IPSec や SSL/TLS で行われるようなセッション鍵の共有や認証のメカニズムは何も規定されていない．したがって，XML 署名と XML 暗号は，IPSec や SSL/TLS とうまく組み合わせて利用することが望ましいであろう．また，XML 署名と XML 暗号を応用して SOAP メッセージ [7]を安全に交換するための仕様が OASIS（Organization for the Advancement of Structured Information Standards）の Web Services Security Technical Committee[8]において標準化されており，今後，XML データをインターネット上で交換するための標準技術の一つとなることが期待されている．

■参考文献

1) W3C Recommendation, Extensible Markup Language (XML) 1.0 (Fourth Edition), 8/16, 2006.
http://www.w3.org/TR/REC-xml
2) W3C Recommendation, XML-Signature Syntax and Processing, 2/12, 2002.
http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
3) W3C Recommendation, XML Encryption Syntax and Processing, 12/10, 2002.
http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
4) W3C Recommendation, Canonical XML Version 1.0, 3/15, 2001.
http://www.w3.org/TR/2001/REC-xml-c14n-20010315
5) W3C Recommendation, XML Path Language (XPath) Version 1.0, 11/16, 1999.
http://www.w3.org/TR/1999/REC-xpath-19991116
6) W3C Recommendation, XSL Transformations (XSLT) Version 1.0, 11/16, 1999.
http://www.w3.org/TR/1999/REC-xslt-19991116
7) W3C XML Protocol Working Group, http://www.w3.org/2000/xp/Group/
8) OASIS, Web Services Security Technical Committee, http://www.oasis-open.org/committees/wss/.
9) W3C Recommendation, XML Encryption Syntax and Processing (Second Edition) , 3/26, 2008.
http://www.w3.org/TR/2008/PER-xmldsig-core-20080326/
10) W3C Recommendation, Canonical XML Version 1.1, 6/21, 2008.
http://www.w3.org/TR/2007/CR-xml-c14n11-20070621/