DATA SECURITY MANAGEMENT

APPLICATION LAYER SECURITY PROTOCOLS FOR NETWORKS

Bill Stackpole

INSIDE

A Layer-by-Layer Look at Security Measures; Interoperability; Standard Security Services; Algorithms Tried and True; Visa's Secure Electronic Transaction Protocol; Securing Electronic Messages; Privacy Enhanced Mail (PEM); Web Application Security; Monetary Transaction Security

WE ARE NOT IN KANSAS ANYMORE

The incredible growth of Internet usage has shifted routine business transactions from fax machine and telephones to e-mail and E-commerce. This shift can be attributed in part to the economical worldwide connectivity of the Internet, but also to the Internet capacity for more sophisticated types of transactions. Security professionals must understand the issues and risks associated with these transactions if they want to provide viable and scalable security solutions for Internet commerce.

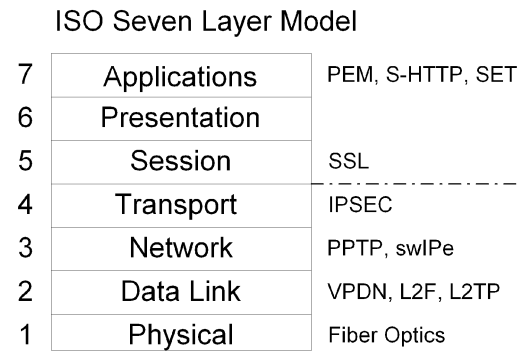
Presence on the Internet makes it possible to conduct international, multiple-party, and multiple-site transactions, regardless of time or language differences. This level of connectivity has, however, created a serious security dilemma for commercial enterprises. How can a company maintain transactional compatibility with thousands of different systems and still ensure the confidentiality of those transactions? Security measures once deemed suitable for text-based messaging and file transfers seem wholly inadequate for sophisticated multimedia and E-commerce transfers. Given the complexity of these transactions, even standardized security protocols like IPsec are proving inadequate.

This article covers three areas that are of particular concern: electronic messaging, World Wide Web (WWW) transactions, and monetary exchanges. All are subject to potential risk of significant financial losses

PAYOFF IDEA

The technologies outlined in this article provide real solutions for mitigating Internet business risks. It is possible to secure messages, Web applications, and monetary exchanges. It would be foolish for the information security officer to not understand and implement the appropriate techniques.

EXHIBIT 1 —



as well as major legal and public relations liabilities. These transactions require security well beyond the capabilities of most lower-layer security protocols. They require application layer security.

A LAYER-BY-LAYER LOOK AT SECURITY MEASURES

Before going into the particulars of application-based security, it may be helpful to look at how security is implemented at the different ISO layers. Exhibit 1 depicts the ISO model divided into upper-layer protocols (those associated with the application of data) and lower-layer protocols (those associated with the transmission of data). Examples of some of the security protocols used at each layer are listed on the right. Begin with layer 1, the physical layer.

Common methods for providing security at the physical layer include:

- securing the cabling conduits: encase them in concrete
- shielding against spurious emissions: TEMPEST
- using media that are difficult to tap: fiber optics

While effective, these methods are limited to things within one's physical control.

Common layer 2 measures include physical address filtering and tunneling (e.g., L2F, L2TP). These measures can be used to control access and provide confidentiality across certain types of connections, but are limited to segments where the endpoints are well-known to the security implementer. Layer 3 measures provide for more sophisticated filtering and tunneling (e.g., PPTP) techniques. Standardized implementations such as IPsec can provide a high degree of security across multiple platforms. However, layer 3 protocols are ill-suited for multiple-site imple-

EXHIBIT 2 — File Transfer Protocol – Command – Packet

Ethernet Header	IP Header	TCP Header	Payload
0040A0…40020A	10.1.2.1…10.2.1.2	FTP (Command)	List…

mentations because they are limited to a single network. Layer 4 transport-based protocols overcome the single network limitation but still lack the sophistication required for multiple-party transactions. Like all lower-layer protocols, transport-based protocols do not interact with the data contained in the payload, so they are unable to protect against payload corruption or content-based attacks.

Application Layer Security: ALS 101

This is precisely the advantage of upper-layer protocols. Application-based security has the capability of interpreting and interacting with the information contained in the payload portion of a datagram. Take, for example, the application proxies used in most firewalls for FTP transfers. These proxies have the ability to restrict the use of certain commands although the commands are contained within the payload portion of the packet. When an FTP transfer is initiated, it sets up a connection for passing commands to the server. The commands one types (e.g., LIST, GET, PASV) are sent to the server in the payload portion of the command packet, as illustrated in Exhibit 2. The firewall proxy — because it is application based — has the ability to "look" at these commands and can therefore restrict their use.

Lower-layer security protocols like IPsec do not have this capability. They can encrypt the commands for confidentiality and authentication, but they cannot restrict their use.

But what exactly is application layer security? As the name implies, it is security provided by the application program itself. For example, a data warehouse using internally maintained access control lists to limit user access to files, records, or fields is implementing application-based security. Applying security at the application level makes it possible to deal with any number of sophisticated security requirements and accommodate additional requirements as they come along. This scenario works particularly well when all applications are contained on a single host or secure intranet, but it becomes problematic when one attempts to extend its functionality across the Internet to thousands of different systems and applications. Traditionally, security in these environments has been addressed in a proprietary fashion within the applications themselves, but this is rapidly changing. The distributed nature of applications on the Internet has given rise to several standardized solutions designed to replace these ad hoc, vendor-specific security mechanisms.

INTEROPERABILITY: THE KEY TO SUCCESS FOR ALS

Interoperability is crucial to the success of any protocol used on the Internet. Adherence to standards is crucial to interoperability. Although the ALS protocols discussed in this article cover three distinctly different areas, they are all based on a common set of standards and provide similar security services. This section introduces some of these common elements. Not all common elements are included, nor are all those covered herein found in every ALS implementation, but there is sufficient commonality to warrant their inclusion.

Cryptography is the key component of all modern security protocols. However, the management of cryptographic keys has in the past been a major deterrent to its use in open environments like the Internet. With the advent of digital certificates and public key management standards, this deterrent has been largely overcome. Standards like the *Internet Public Key Infrastructure X.509* (pkix) and the *Simple Public Key Infrastructure* (spki) provide the mechanisms necessary to issue, manage, and validate cryptographic keys across multiple domains and platforms. All of the protocols discussed in this article support the use of this public key infrastructure.

STANDARD SECURITY SERVICES: MAXIMUM MESSAGE PROTECTION

All the ALS protocols discussed in this article provide the following four standard security services:

- *confidentiality* (a.k.a. privacy): the assurance that only the intended recipient can read the contents of the information sent to them
- *integrity*: the guarantee that the information received is exactly the same as the information that was sent
- *authentication:* the guarantee that the sender of a message or transmission is really who he or she claims to be
- *nonrepudiation:* the proof that a message was sent by its originator, even if the originator claims it was not

Each of these services relies on a form of cryptography for its functionality. Although the service implementations may vary, they all use a fairly standard set of algorithms.

ALGORITHMS TRIED AND TRUE

The strength of a cryptographic algorithm can be measured by its longevity. Good algorithms continue to demonstrate high cryptographic strength after years of analysis and attack. The ALS protocols discussed here support three types of cryptography — symmetric, asymmetric, and hashing — using time-tested algorithms.

Symmetric (also called secret key) cryptography is primarily used for confidentiality functions because it has high cryptographic strength and

can quickly process large volumes of data. In ALS implementations, DES is the most commonly supported symmetric algorithm. Asymmetric or public-key cryptography is most commonly used in ALS applications to provide confidentiality during the initialization or set-up portion of a transaction. Public keys and digital certificates are used to authenticate the participating parties to one another and exchange the symmetric keys used for the remainder of the transaction. The most commonly supported asymmetric algorithm in ALS implementations is RSA.

Cryptographic hashing is used to provide integrity and authentication in ALS implementations. When used separately, authentication validates the sender and the integrity of the message, but using them in combination provides proof that the message was not forged and therefore cannot be refuted (nonrepudiation). The three most commonly used hashes in ALS applications are MD2, MD5, and SHA. In addition to a common set of algorithms, systems wishing to interoperate in an open environment must be able to negotiate and validate a common set of security parameters. The next section introduces some of the standards used to define and validate these parameters.

STANDARDIZED GIBBERISH IS STILL GIBBERISH

For applications to effectively exchange information, they must agree on a common format for that information. Security services, if they are to be trustworthy, require all parties to function in unison. Communication parameters must be established; security services, modes, and algorithms agreed upon, and cryptographic keys exchanged and validated. To facilitate these processes, the ALS protocols discussed in this article support the following formatting standards:

- X.509: The X.509 Standard defines the format of digital certificates used by Certification Authorities to validate public encryption keys.
- PKCS: The Public Key Cryptography Standard defines the underlying parameters (object identifiers) used to perform the cryptographic transforms and to validate keying data.
- CMS: The Cryptographic Message Syntax defines the transmission formats and cryptographic content types used by the security services. CMS defines six cryptographic content types, ranging from no security to signed and encrypted content. They are data, signedData, envelopedData, signedAndEnvelopedData, digestData, and encryptedData.
- MOSS: The MIME Object Security Services defines two additional cryptographic content types for multi-part MIME (multimedia Internet mail extensions) objects that can be used singularly or in combination. They are multi-part signed and multi-part encrypted.

Encryption is necessary to ensure transaction confidentiality and integrity on open networks, and the Public Key/Certification Authority archi-

tecture provides the infrastructure necessary to manage the distribution and validation of cryptographic keys. Security mechanisms at all levels now have a standard method for initiating secure transactions, thus eliminating the need for proprietary solutions to handle secure multiple-party, multiple-site, or international transactions. A case in point is the new SET credit card transaction protocol.

*SET*TING THE EXAMPLE: VISA'S SECURE ELECTRONIC TRANSACTION PROTOCOL

SET (Secure Electronic Transaction) is an application-based security protocol jointly developed by Visa and MasterCard. It was created to provide secure credit card payment transactions over open networks. SET is the electronic equivalent of a face-to-face or mail order credit card transaction. It provides confidentiality and integrity for payment transmissions and authenticates all parties involved in the transaction. One can take a walk through a SET transaction to see how this application layer protocol handles a sophisticated multi-party financial transaction.

A SET transaction involves five different participants: the Cardholder, the Issuer of the payment card, the Merchant, the Acquirer that holds the merchant's account, and a Payment Gateway that processes SET transactions on behalf of the Acquirer. The policies governing how transactions are conducted are established by a sixth party, the Brand (i.e., Visa), but they do not participate in payment transactions.

A SET transaction requires two pairs of asymmetric encryption keys and two digital certificates — one pair for exchanging information and the other for digital signatures. The keys and certificates can be stored on a "smart" credit card or embedded into any SET-enabled application (i.e., Web browser). The keys and certificates are issued to the Cardholder by a Certification Authority (CA) on behalf of the Issuer. The Merchant's keys and digital certificates are issued to them by a CA on behalf of the Acquirer. They provide assurance that the Merchant has a valid account with the Acquirer. The Cardholder and Merchant certificates are digitally signed by the issuing financial institution to ensure their authenticity and to prevent them from being fraudulently altered. One interesting feature of this arrangement is that the Cardholder's certificate does not contain their account number or expiration date. That information is encoded using a secret key that is only supplied to the Payment Gateway during the payment authorization. Now that all the players are identified, one can get started.

Step 1: The *Cardholder* goes shopping, selects merchandise, and sends a purchase order to the *Merchant* requesting a SET payment type. (The SET specification does not define how shopping is accomplished so it has no involvement in this portion of the transaction.) The *Cardholder* and *Merchant*,

if they have not already, authenticate themselves to each other by exchanging certificates and digital signatures. During this exchange, the *Merchant* also supplies the *Payment Gateway's* certificate and digital signature information to the *Cardholder.* One will see later how this is used. Also established in this exchange is a pair of randomly generated symmetric keys that will be used to encrypt the remaining *Cardholder–Merchant* transmissions.

Step 2: Once the above exchanges have been completed, the *Merchant* contacts the *Payment Gateway.* Part of this exchange includes language selection information to ensure international interoperability. Once again, certificate and digital signature information is used to authenticate the *Merchant* to the *Payment Gateway* and establish random symmetric keys. Payment information (PI) is then forwarded to the *Gateway* for payment authorization. Notice that *only* the payment information is forwarded; this is done to satisfy regulatory requirements regarding the use of strong encryption. Generally, the use of strong cryptography by financial institutions is not restricted if the transaction only contains monetary values.

Step 3: Upon receipt of the PI, the *Payment Gateway* authenticates the *Cardholder.* Notice that the *Cardholder* is authenticated without directly contacting the *Purchase Gateway.* This is done through a process called dual digital signature. The information required by the *Purchase Gateway* to authenticate the *Cardholder* is sent to the *Merchant* with a different digital signature than the one used for *Merchant–Cardholder* exchanges. This is possible because the *Merchant* sent the *Purchase Gateway* certificates to the *Cardholder* in an earlier exchange. The *Merchant* simply forwards this formation to the *Payment Gateway* as part of the payment authorization request. Another piece of information passed in this exchange is the secret key the *Gateway* needs to decrypt the *Cardholder's* account number and expiration date.

Step 4: The *Gateway* reformats the payment information and forwards it via a private circuit to the *Issuer* for authorization. When the *Issuer* authorizes the transaction, the *Payment Gateway* notifies the *Merchant,* who notifies the *Cardholder,* and the transaction is complete.

Step 5: The *Merchant* finalizes the transaction by issuing a Payment Capture request to the *Payment Gateway,* causing the *Cardholder's* account to be debited, and the *Merchant's* account to be credited for the transaction amount.

A single SET transaction like the one outlined above is incredibly complex, requiring more than 59 different actions to take place successfully. Such complexity requires application layer technology to be managed effectively. The beauty of SET, however, is its ability to do just that in a secure and ubiquitous manner. Other protocols are achieving similar success in different application areas.

FROM POSTCARDS TO LETTERS: SECURING ELECTRONIC MESSAGES

Electronic messaging is a world of postcards. As messages move from source to destination, they are openly available (like writing on a postcard) to be read by those handling them. If postcards are not suitable for business communications it stands to reason that electronic mail on an open network is also not suitable. Standard business communications, require confidentiality, and other, more-sensitive communications require additional safeguards like proof of delivery or sender verification, features that are not available in the commonly used Internet mail protocols. This has led to the development of several security-enhanced messaging protocols. PEM is one such protocol.

Privacy Enhanced Mail (PEM)

PEM is an application layer security protocol developed by the IETF (Internet Engineering Task Force) to add confidentiality and authentication services to electronic messages on the Internet. The goal was to create a standard that could be implemented on any host, be compatible with existing mail systems, support standard key management schemes, protect both individually addressed and list addressed mail, and not interfere with non-secure mail delivery. When the standard was finalized in 1993, it had succeeded on all counts. PEM supports all four standard security services, although all services are not necessarily part of every message. PEM messages can be MIC-CLEAR messages that provide integrity and authentication only; MIC-ONLY messages that provide integrity and authentication with support for certain gateway implementations; or ENCRYPTED messages that provides integrity, authentication, and confidentiality.

Some key PEM features include:

- End-to-end confidentiality: messages are protected against disclosure from the time they leave the sender's system until they are *read* by the recipient.
- Sender and forwarder authentication: PEM digital signatures authenticate both senders and forwarders and ensure message integrity. PEM utilizes an integrity check that allows messages to be received in any order and still be verified — an important feature in environments like the Internet where messages can be fragmented during transit.

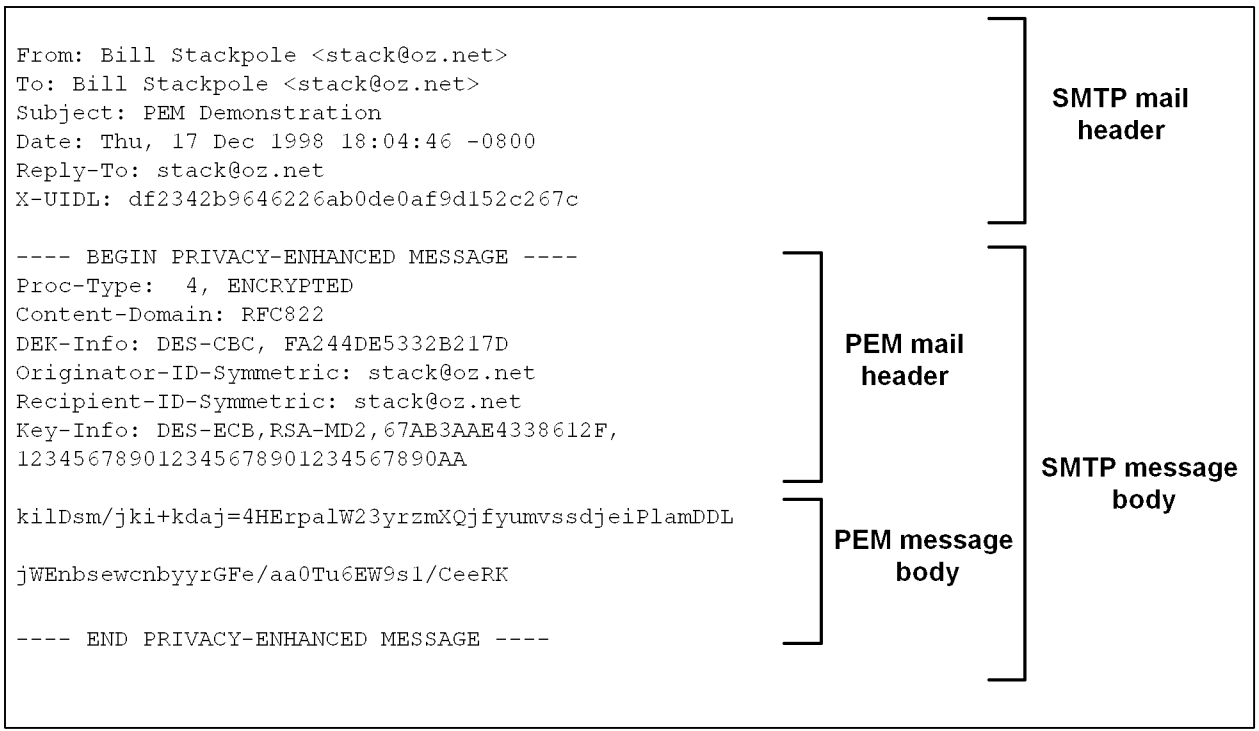
- Originator nonrepudiation: this feature authenticates the *originator* of a PEM message. It is particularly useful for forwarded messages because a PEM digital signature only authenticates the last sender. Nonrepudiation verifies the originator no matter how many times the message is forwarded.
- Algorithm independence: PEM was designed to easily accommodate new cryptographic and key management schemes. Currently, PEM supports common algorithms in four areas: DES for data encryption, DES and RSA for key management, RSA for message integrity, and RSA for digital signatures.
- PKIX support: PEM fully supports interoperability on open networks using the Internet Public Key Infrastructure X.509.
- Delivery system independence: PEM achieves delivery system independence because its functions are contained in the body of a standard message and use a standard character set, as illustrated in Exhibit 3.
- X.500 distinguished name support: PEM uses the distinguished name (DN) feature of the X.500 directory standard to identify senders and recipients. This feature separates mail from specific individuals, allowing organizations, lists, and systems to send and receive PEM messages.

RIPEM (Riordan's Internet Privacy Enhanced Mail) is public domain implementation of the PEM protocol, although not in its entirety. Since the author, Mark Riordan, placed the code in the public domain, it has been ported to a large number of operating systems. Source and binaries are available via FTP to U.S. and Canadian citizens from ripem.msu.edu. Read the GETTING_ACCESS file in the /pub/crypt/ directory before attempting any downloads.

Secure/Multi-purpose Internet Mail Extensions (S/MIME)

S/MIME is another application layer protocol that provides all four standard security services for electronic messages. Originally designed by RSA Data Security, the S/MIME specification is currently managed by the IETF S/MIME Working Group. Although S/MIME is not an IETF standard, it has already garnered considerable vendor support, largely because it is based on well-proven standards that provide a high degree of interoperability. Most notable is, of course, the popular and widely used MIME standard, but S/MIME also utilizes the CMS, PKCS, and X.509 standards. Like PEM, S/MIME is compatible with most existing Internet mail systems and does not interfere with the delivery of non-secure messages. However, S/MIME has the added benefit of working seamlessly with other MIME transports (i.e., HTTP) and can even function in mixed-transport environments. This makes it particularly attractive for use with automated transfers like EDI and Internet fax.

EXHIBIT 3 —



There are two S/MIME message types: signed, and signed and enveloped. Signed messages provide integrity and sender authentication, while signed and enveloped messages provide integrity, authentication, and confidentiality. The remaining features of S/MIME are very similar to PEM and do not warrant repeating here.

A list of commercial S/MIME products that have successfully completed S/MIME interoperability testing is available on the RSA Data Security Web site at www.rsa.com/smime/html/interop_center.html. A public domain version of S/MIME, written in PERL by Ralph Levien, is available at www.c2.org/~raph/premail.html.

Open Pretty Good Privacy (OpenPGP)

OpenPGP, sometimes called PGP/MIME, is another emerging ALS protocol on track to becoming an IETF standard. It is based on PGP, the most widely deployed message security program on the Internet. OpenPGP is very similar in features and functionality to S/MIME, but the two protocols are not interoperable because they use slightly different encryption algorithms and MIME encapsulations. A list of PGP implementations and other OpenPGP information is available at http://www-ns.rutgers.edu/~mione/openpgp/. Freeware implementations of OpenPGP are available from the North American Cryptography Archives (www.cryptography.org).

TAMING HTTP: WEB APPLICATION SECURITY

Web-based applications are quickly becoming the standard for all types of electronic transactions because they are easy to use and highly interoperable. These features are also their major security failing. Web transactions traverse the network in well-known and easily intercepted formats, making them quite unsuitable for most business transactions. This section discusses some of the mechanisms used to overcome these Web security issues.

Secure HyperText Transfer Protocol (S/HTTP)

S/HTTP is a message-oriented security protocol designed to provide end-to-end confidentiality, integrity, authentication, and nonrepudiation services for HTTP clients and servers. It was originally developed by Enterprise Integration Technologies (now Verifone, Inc.) in 1995. At this writing, S/HTTP is still an IETF draft standard, but it is already widely used in Web applications. Its success can be attributed to a flexible design that is rooted in established standards. The prominent standard is, of course, HTTP, but the protocol also utilizes the NIST Digital Signature Standard (DSS), CMS, MOSS, and X.509 standards. Strict adherence of S/HTTP to the HTTP messaging model provides delivery system independence and

makes it easy to integrate S/HTTP functions into standard HTTP applications. Algorithm independence and the ability to negotiate security options between participating parties ensures the interoperability of S/HTTP for years to come. Secure HTTP modes of operation include message protection, key management, and a transaction freshness mechanism.

Secure HTTP protection features include:

- Support for MOSS and CMS: protections are provided in both content domains using the CMS "application/s-http" content-type or the MOSS "multipart-signed" or "multipart-encrypted" header.
- Syntax compatibility: protection parameters are specified by extending the range of HTTP message headers, making S/HTTP messages syntactically the same as standard HTTP messages, except that the range of the headers is different and the body is usually encrypted.
- Recursive protections: protections can be used singularly or applied one layer after another to achieve higher levels of protection. Layering the protections makes it easier for the receiving system to parse them. The message is simply parsed one protection at a time until it yields a standard HTTP content type.
- Algorithm independence: the S/HTTP message structure can easily incorporate new cryptographic implementations. The current specification requires supporting MD5 for message digests, MD5-HMAC for authentication, DES-CBC for symmetric encryption, and NIST-DSS for signature generation and verification.
- Freshness feature: S/HTTP uses a simple challenge-response to ensure that the data being returned to the server is "fresh." In environments like HTTP where long periods of time can pass between messages, it is difficult to track the state of a transaction. To overcome this problem, the originator of an HTTP message sends a freshness value (nonce) to the recipient along with the transaction data. The recipient returns the nonce with the response. If the nonces match, the data is fresh and the transaction can continue. Stale data indicates an error condition.

S/HTTP key management modes include:

- Manual exchange: shared secrets are exchanged through a simple password mechanism like PAP. The server simply sends the client a dialog box requesting a userID and password, and then authenticates the response against an existing list of authorized users.
- Public key exchange: keys are exchanged using the Internet Public Key Infrastructure with full X.509 certificate support. S/HTTP implementations are required to support Diffie-Hellman for in-band key exchanges.

- Out-of-band key exchange: symmetric keys can be prearranged through some other media (e.g., snail mail). This feature, unique to the S/HTTP, permits parties that do not have established public keys to participate in secure transactions.
- In-band symmetric key exchange: S/HTTP can use public-key encryption to exchange random symmetric keys in instances where the transaction would benefit from the higher performance of symmetric encryption.

Many commercial Web browsers and servers implement the S/HTTP protocol, but the author was unable to find any public domain implementations. A full implementation of S/HTTP including the C source code is available in the SecureWeb Toolkit™ from Terisa (www.spyrus.com). The kit also contains the source code for SSL.

Secure Socket Layer (SSL)

SSL is a client/server protocol designed by Netscape to provide secure communications for its Web browser and server products. It was quickly adopted by other vendors and has become the *de facto* standard for secure Web transactions. However, SSL is not limited to Web services; it can provide confidentiality, integrity, authentication, and nonrepudiation services between any two communicating applications. The current version of SSL (SSL V3.0) is on track to become an IETF standard. While included here as an application layer protocol, SSL is actually designed to function at the session and application layers. The SSL Record Protocol provides security services at the session layer — the point where the application interfaces to the TCP/IP transport sockets. It is used to encapsulate higher-layer protocols and data for compression and transmission. The SSL Handshake Protocol is an application-based service used to authenticate the client and server to each other and negotiate the security parameters for each communication session.

The SSL Handshake Protocol utilizes public-key encryption with X.509 certificate validation to negotiate the symmetric encryption parameters used for each client/server session. SSL is a stateful protocol. It transitions through several different states during connection and session operations. The Handshake Protocol is used to coordinate and maintain these states. One SSL session may include multiple connections, and participating parties may have multiple simultaneous sessions. The session state maintains the peer certificate information, compression parameters, cipher parameters, and the symmetric encryption key. The connection state maintains the MAC and asymmetric keys for the client and server, as well as the vectors (if required) for symmetric encryption initialization. SSL was designed to be fully extensible and can support multiple encryption schemes. The current version requires support for the following schemes:

- DES, RC2, RC4, and IDEA for confidentiality
- RSA and DSS for peer authentication
- SHA and MD5 for message integrity
- X.509 and FORTEZZA certificates for key validation
- RSA, Diffie-Hellman, and FORTEZZA for key exchange

SSL also supports NULL parameters for unsigned and unencrypted transmissions. This allows the implementer to apply an appropriate amount of security to their application. The support for the FORTEZZA hardware encryption system is unique to SSL, as is the data compression requirement. SSL uses a session caching mechanism to facilitate setting up multiple sessions between clients and servers and resuming disrupted sessions.

There is an exceptional public domain implementation of SSL, created by Eric Young and Tim Hudson of Australia, called SSLeay. It includes a full implementation of Netscapes's SSL version 2 with patches for telnet, FTP, Mosaic, and several Web servers. The current version is available from the SSLeay Web site at www.ssleay.org. The site includes several SSL white papers and an excellent *Programmers Reference.*

DO NOT SHOW ME THE MONEY: MONETARY TRANSACTION SECURITY

The success of commerce on the Internet depends on its ability to conduct monetary transactions securely. Although purchasing seems to dominate this arena, bill payment, fund and instrument transfers, and EDI are important considerations. The lack of standards for electronic payment has fostered a multitude of proprietary solutions, including popular offerings from Cybercash (Cybercoin), Digital (Millicent), and Digicash. However, proprietary solutions are not likely to receive widespread success in a heterogeneous environment like the Internet. This section concentrates on standardized solutions. Because the SET Protocol has been covered in some detail already, only SET implementations are mentioned here.

Secure Payment (S/PAY)

S/PAY is a developer's toolkit based on the SET protocol. It was developed by RSA Data Security, although the marketing rights currently belong to the Trintech Group (www.trintech.com). The S/PAY library fully implements the SET v1.0 cardholder, merchant, and acquirer functions and the underlying encryption and certificate management functions for Windows 95/NT and major UNIX platforms. Included in the code is support for hardware-based encryption engines, smart card devices, and long-term private-key storage. Trintech also offers full implementations of SET merchant, cardholder, and acquirer software. This includes their PayWare Net-POS product, which supports several combinations of SSL and SET technologies aimed at easing the transition from Web SSL transactions to fully implemented SET transactions.

Open Financial Exchange (OFX)

OFX is an application layer protocol created by Checkfree, Intuit, and Microsoft to support a wide range of consumer and small-business banking services over the Internet. OFX is an open specification available to any financial institution or vendor desiring to implement OFX services. OFX uses SSL with digital certificate support to provide confidentiality, integrity, and authentication services to its transactions. The protocol has gained considerable support in the banking and investment industry because it supports just about every conceivable financial transaction. Currently, the OFX committee is seeking to expand presence of OFX through interoperability deals with IBM and other vendors. Copies of the OFX specification are available from the Open Financial Exchange Web site (www.ofx.net).

Micro Payment Transfer Protocol (MPTP)

MPTP is part of the World Wide Web Consortium (W3C) Joint Electronic Payment Initiative. Currently, MPTP is a W3C working draft. The specification is based on variations of Rivest and Shamir's Pay-Word, Digital's Millicent, and Bellare's iKP proposals. MPTP is a very flexible protocol that can be layered upon existing transports like HTTP or MIME to provide greater transaction scope. It is highly tolerant of transmission delays, allowing much of the transaction processing to take place offline. MPTP is designed to provide payments through the services of a third-party broker. In the current version, the broker must be common to both the customer and the vendor, although inter-broker transfers are planned for future implementations. This will be necessary if MPTP is going to scale effectively to meet Internet demands.

Customers establish an account with a broker. Once established, they are free to purchase from any vendor common to their broker. The MPTP design takes into consideration the majority of risks associated with electronic payment and provides mechanisms to mitigate those risks, but it does not implement a specific security policy. Brokers are free to define policies that best suit their business requirements.

MPTP relies on S/Key technology using the MD5 or SHA algorithm to authorize payments. MPTP permits the signing of messages for authentication, integrity, and nonrepudiation using public- or secret-key cryptography and fully supports X.509 certificates. Although MPTP is still in the draft stages, its exceptional design, flexibility, and high performance destine it to be a prime contender in the electronic payment arena.

Java Electronic Commerce Framework (JECF)

JECF is the final item of discussion. JECF is not an application protocol. It is a framework for implementing electronic payment processing using active content technology. Active content technology uses an engine (i.e,

a Java virtual machine) installed on the client to execute program components (e.g., applets) sent to it from the server. Current JECF active content components include Java Commerce Messages, Gateway Security Model, Commerce JavaBeans, and Java Commerce Client (JCC).

JECF is based on the concept of an electronic wallet. The wallet is an extensible client-side mechanism capable of supporting any number of E-commerce transactions. Vendors create Java applications consisting of service modules (applets) called Commerce JavaBeans that plug in to the wallet. These applets implement the operations and protocols (i.e., SET) necessary to conduct transactions with the vendor. There are several significant advantages to this architecture, including:

- Vendors are not tied to specific policies for their transactions. They are free to create modules containing policies and procedures best suited to their business.
- Clients are not required to have specialized applications. Since JavaBean applets are active content, they can be delivered and dynamically loaded on the customer's system as the transaction is taking place.
- Applications can be updated dynamically. Transaction applets can be updated or changed to correct problems or meet growing business needs without having to send updates out to all the clients. The new modules will be loaded over the old during their next transaction.
- Modules can be loaded or unloaded on-the-fly to accommodate different payment, encryption, or language requirements. OFX modules can be loaded for banking transactions and later unloaded when the customer requires SET modules to make a credit card purchase.
- JavaBean modules run on any operating system, browser, or application supporting Java. This gives vendors immediate access to the largest possible customer base.

The flexibility, portability, and large Java user base make the JECF a very attractive E-commerce solution. It is sure to become a major player in the electronic commerce arena.

IF IT IS NOT ENCRYPTED NOW ...

The Internet has dramatically changed the way business is done, but that has not come without a price. Security for Internet transactions and messaging is woefully lacking, making much of what is being done on the Internet an open book for all to read. This cannot continue. Despite the complexity of the problems facing us, there are solutions. The technologies outlined in this article provide real solutions for mitigating Internet business risks. One can secure messages, Web applications, and monetary exchanges. Admittedly, some of these applications are not as pol-

ished as one would like, and some are difficult to implement and manage, but they are nonetheless effective and most certainly a step in the right direction.

Someday, all business transactions on the Internet will be encrypted, signed, sealed, and delivered, but the author is not sure if we can wait for that day. Business transactions on the Internet are increasing and new business uses for the Internet are going to be found. Waiting for things to get better is only going to put us further behind the curve. Someone has let the Internet bull out of the cage and we are either going to take him by the horns or get run over! ALS now!

References

1. *E-mail Security: How to Keep Your Electronic Messages Private* by Bruce Schneier, John Wiley & Sons, 1995
2. *Internet Draft — The Secure HyperText Transfer Protocol*, E. Resorla and A. Schiffman, Terisa Systems, Inc., June 1998
3. *A Review of E-mail Security Standards*, Laurence Lundblade, Qualcomm, Inc., 1998.
4. *SET Secure Electronic Transaction Specification, Book 1: Business Description*, Setco, Inc., May 31, 1997.
5. S/MIME: Anatomy of a Secure E-mail Standard, Steve Dusse and Tim Matthews, *Messaging Magazine*, 1998.
6. *Internet Draft — S/MIME Version 3 Message Specification*, Blake Ramsdell, Worldtalk, Inc., August 6, 1998.
7. *RFC 1848 — MIME Object Security Services*, S. Crocker, N. Freed, J. Galvan, and S. Murphy, IETF, October, 1995.
8. *PKCS #7: Cryptographic Message Syntax Standard*, RSA Laboratories Technical Note Version 1.5, November 1, 1993, © RSA Laboratories.
9. *Micro Payment Transfer Protocol (MPTP) Version 1.0*, Phillip Hallam-Baker, Joint Electronic Payment Initiative–W3C, November, 1995.
10. *Micropayments*, David Pearah, Massachusetts Institute of Technology, April 23, 1997.
11. *Internet Draft — The SSL Protocol Version 3.0*, Alan O. Freier, Philip Karlton, and Paul C. Kocher, November 18, 1996.
12. *SSL Programmers Reference*, T.J. Hudson and E.A. Young, July 1, 1995.
13. *Introducing SSL and Certificates Using SSLeay*, Frederick Hirsch, The Open Group Research Institute, *World Wide Web Journal*, Summer, 1997.

Notes

1. The Internet Mail Consortium (IMC), www.inc.org.
2. The Electronic Messaging Association, www.ema.org.
3. S/MIME Central, http://www.rsa.com/smime/.
4. Information Society Project Office (ISPO), www.ispo.cec.be.
5. Java Commerce Products, http://java.sun.com.
6. E-Payments Resource Center, Trintech Inc., www.trintech.com.
7. Transaction Net and the Open Financial Exchange, www.ofx.net.
8. SET Reference Implementation (SETREF), Terisa Inc., www.terisa.com.
9. SET — Secure Electronic Transaction LLC, www.setco.org.

Bill Stackpole works for Olympic Resource Management in Poulsbo, WA.