

DANE = DNS-based Authentication of Named Entities
 DKIM = DomainKeys Identified Mail
 DMARC = Domain-based Message Authentication, Reporting, and Conformance
 DNSSEC = Domain Name System Security Extensions
 SPF = Sender Policy Framework
 S/MIME = Secure Multi-Purpose Internet Mail Extensions
 TLSA RR = Transport Layer Security Authentication Resource Record

Figure 19.4 The Interrelationship of DNSSEC, SPF, DKIM, DMARC, DANE, and S/MIME for Assuring Message Authenticity and Integrity

Figure 19.4 shows how these components interact to provide message authenticity and integrity. Not shown, for simplicity, is that S/MIME also provides message confidentiality by encrypting messages.

19.4 S/MIME

Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet email format standard based on technology from **RSA Data Security**. S/MIME is a complex capability that is defined in a number of documents. The most important documents relevant to S/MIME include the following:

- **RFC 5750, S/MIME Version 3.2 Certificate Handling:** Specifies conventions for X.509 certificate usage by (S/MIME) v3.2.

- **RFC 5751, S/MIME) Version 3.2 Message Specification:** The principal defining document for S/MIME message creation and processing.
- **RFC 4134, Examples of S/MIME Messages:** Gives examples of message bodies formatted using S/MIME.
- **RFC 2634, Enhanced Security Services for S/MIME:** Describes four optional security service extensions for S/MIME.
- **RFC 5652, Cryptographic Message Syntax (CMS):** Describes the Cryptographic Message Syntax (CMS). This syntax is used to digitally sign, digest, authenticate, or encrypt arbitrary message content.
- **RFC 3370, CMS Algorithms:** Describes the conventions for using several cryptographic algorithms with the CMS.
- **RFC 5752, Multiple Signatures in CMS:** Describes the use of multiple, parallel signatures for a message.
- **RFC 1847, Security Multiparts for MIME—Multipart/Signed and Multipart/Encrypted:** Defines a framework within which security services may be applied to MIME body parts. The use of a digital signature is relevant to S/MIME, as explained subsequently.

Operational Description

S/MIME provides for four message-related services: **authentication**, **confidentiality**, **compression**, and **email compatibility** (Table 19.4). This subsection provides an overview. We then look in more detail at this capability by examining message formats and message preparation.

AUTHENTICATION Authentication is provided by means of a **digital signature**, using the general scheme discussed in Chapter 13 and illustrated in Figure 13.1. Most commonly RSA with SHA-256 is used. The sequence is as follows:

1. The **sender** creates a message.
2. **SHA-256** is used to generate a **256-bit message digest** of the message.

Table 19.4 Summary of S/MIME Services

Function	Typical Algorithm	Typical Action
Digital signature	RSA/SHA-256	A hash code of a message is created using SHA-256. This message digest is encrypted using SHA-256 with the sender's private key and included with the message.
Message encryption	AES-128 with CBC	A message is encrypted using AES-128 with CBC with a one-time session key generated by the sender. The session key is encrypted using RSA with the recipient's public key and included with the message.
Compression	unspecified	A message may be compressed for storage or transmission .
Email compatibility	Radix-64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

3. The message digest is encrypted with **RSA** using the **sender's private key**, and the result is appended to the message. Also **appended** is identifying information for the signer, which will enable the receiver to retrieve the **signer's public key**.
4. The **receiver uses RSA with the sender's public key to decrypt and recover the message digest**.
5. The receiver generates a new message digest for the message and compares it with the **decrypted hash code**. If the two match, the message is accepted as authentic.

The combination of SHA-256 and RSA provides an effective digital signature scheme. Because of the strength of RSA, the recipient is assured that only the possessor of the matching private key can generate the signature. Because of the strength of SHA-256, the recipient is assured that no one else could generate a new message that matches the hash code and, hence, the signature of the original message.

Although signatures normally are found attached to the message or file that they sign, this is not always the case: Detached signatures are supported. A detached signature may be stored and transmitted separately from the message it signs. This is useful in several contexts. A user may wish to maintain a separate signature log of all messages sent or received. A detached signature of an executable program can detect subsequent virus infection. Finally, detached signatures can be used when more than one party must sign a document, such as a legal contract. Each person's signature is independent and therefore is applied only to the document. Otherwise, signatures would have to be nested, with the second signer signing both the document and the first signature, and so on.

CONFIDENTIALITY S/MIME provides confidentiality by encrypting messages. Most commonly **AES** with a **128-bit key** is used, with the **cipher block chaining (CBC)** mode. The key itself is also encrypted, typically with RSA, as explained below.

As always, one must address the problem of key distribution. In S/MIME, each symmetric key, referred to as a **content-encryption key**, is **used only once**. That is, a new key is generated as a random number for each message. Because it is to be used only once, the content-encryption key is bound to the message and transmitted with it. To **protect the key**, it is encrypted with the **receiver's public key**. The sequence can be described as follows:

1. The sender generates a message and a random 128-bit number to be used as a content-encryption key for this message only.
2. The message is encrypted using the content-encryption key.
3. The content-encryption key is encrypted with RSA using the recipient's public key and is attached to the message.
4. The receiver uses RSA with its private key to decrypt and recover the content-encryption key.
5. The content-encryption key is used to decrypt the message.

Several observations may be made. First, to reduce encryption time, the combination of symmetric and public-key encryption is used in preference to simply using public-key encryption to encrypt the message directly: Symmetric algorithms

are substantially faster than asymmetric ones for a large block of content. Second, the use of the public-key algorithm solves the session-key distribution problem, because only the recipient is able to recover the session key that is bound to the message. Note that we do not need a session-key exchange protocol of the type discussed in Chapter 14, because we are not beginning an ongoing session. Rather, each message is a one-time independent event with its own key. Furthermore, given the store-and-forward nature of electronic mail, the use of handshaking to assure that both sides have the same session key is not practical. Finally, the use of one-time symmetric keys strengthens what is already a strong symmetric encryption approach. Only a small amount of plaintext is encrypted with each key, and there is no relationship among the keys. Thus, to the extent that the public-key algorithm is secure, the entire scheme is secure.

CONFIDENTIALITY AND AUTHENTICATION As Figure 19.5 illustrates, both confidentiality and encryption may be used for the same message. The figure shows a sequence in which a signature is generated for the plaintext message and appended to the message. Then the plaintext message and signature are encrypted as a single block using symmetric encryption and the symmetric encryption key is encrypted using public-key encryption.

S/MIME allows the signing and message encryption operations to be performed in either order. If signing is done first, the identity of the signer is hidden by the encryption. Plus, it is generally more convenient to store a signature with a plaintext version of a message. Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature.

If encryption is done first, it is possible to verify a signature without exposing the message content. This can be useful in a context in which automatic signature verification is desired, as no private key material is required to verify a signature. However, in this case the recipient cannot determine any relationship between the signer and the unencrypted content of the message.

EMAIL COMPATIBILITY When S/MIME is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key). Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, S/MIME provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters, a process referred to as 7-bit encoding.

The scheme typically used for this purpose is Base64 conversion. Each group of three octets of binary data is mapped into four ASCII characters. See Appendix X for a description.

One noteworthy aspect of the Base64 algorithm is that it blindly converts the input stream to Base64 format regardless of content, even if the input happens to be ASCII text. Thus, if a message is signed but not encrypted and the conversion is applied to the entire block, the output will be unreadable to the casual observer, which provides a certain level of confidentiality.

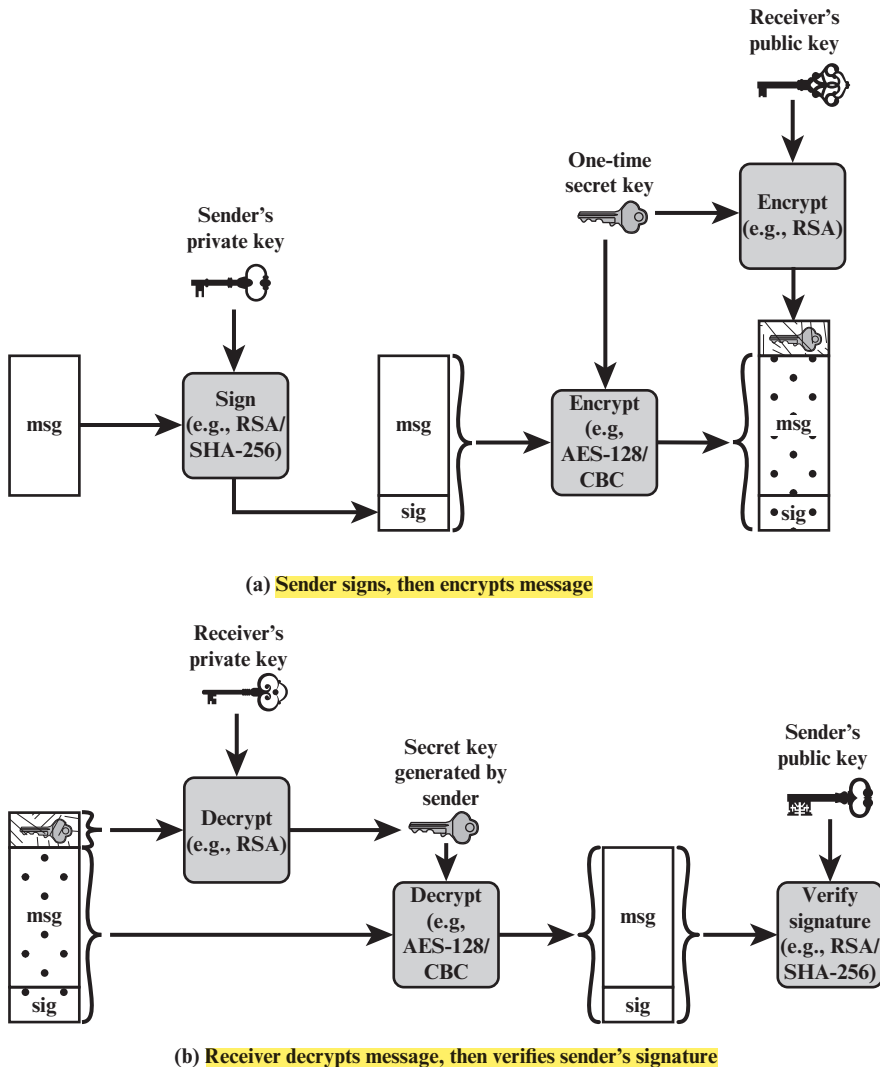


Figure 19.5 Simplified S/MIME Functional Flow

RFC 5751 also recommends that even if outer 7-bit encoding is not used, the original MIME content should be 7-bit encoded. The reason for this is that it allows the MIME entity to be handled in any environment without changing it. For example, a trusted gateway might remove the encryption, but not the signature, of a message, and then forward the signed message on to the end recipient so that they can verify the signatures directly. If the transport internal to the site is not 8-bit clean, such as on a wide area network with a single mail gateway, verifying the signature will not be possible unless the original MIME entity was only 7-bit data.

COMPRESSION S/MIME also offers the ability to **compress a message**. This has the benefit of **saving space both for email transmission and for file storage**. Compression

can be applied in any order with respect to the signing and message encryption operations. RFC 5751 provides the following guidelines:

- **Compression of binary encoded encrypted data is discouraged**, since it will not yield significant compression. Base64 encrypted data could very well benefit, however.
- If a lossy compression algorithm is used with signing, you will need to compress first, then sign.

S/MIME Message Content Types

S/MIME uses the following message content types, which are defined in RFC 5652, Cryptographic Message Syntax:

- **Data:** Refers to the **inner MIME-encoded message content**, which may then be encapsulated in a SignedData, EnvelopedData, or CompressedData content type.
- **SignedData:** Used to **apply a digital signature** to a message.
- **EnvelopedData:** This consists of **encrypted content** of any type and encrypted-content **encryption keys** for one or more recipients.
- **CompressedData:** Used to **apply data compression to a message**.

The Data content type is also used for a procedure known as **clear signing**. For clear signing, a digital signature is calculated for a **MIME-encoded message** and the two parts, the message and signature, form a multipart MIME message. Unlike SignedData, which involves encapsulating the message and signature in a special format, clear-signed messages can be read and their signatures verified by email entities that do not implement S/MIME.

Approved Cryptographic Algorithms

Table 19.5 summarizes the cryptographic algorithms used in S/MIME. S/MIME uses the following terminology taken from RFC 2119 (*Key Words for use in RFCs to Indicate Requirement Levels*, March 1997) to specify the requirement level:

- **MUST:** The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.
- **SHOULD:** There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

The S/MIME specification includes a discussion of the procedure for deciding which content encryption algorithm to use. In essence, a sending agent has two decisions to make. First, the sending agent must determine if the receiving agent is capable of decrypting using a given encryption algorithm. Second, if the receiving agent is only capable of accepting weakly encrypted content, the sending agent must decide if it is acceptable to send using weak encryption. To support this decision process, a sending agent may announce its decrypting capabilities in order of preference for any message that it sends out. A receiving agent may store that information for future use.

Table 19.5 Cryptographic Algorithms Used in S/MIME

Function	Requirement
Create a message digest to be used in forming a digital signature.	MUST support SHA-256 SHOULD support SHA-1 Receiver SHOULD support MD5 for backward compatibility
Use message digest to form a digital signature.	MUST support RSA with SHA-256 SHOULD support — DSA with SHA-256 — RSASSA-PSS with SHA-256 — RSA with SHA-1 — DSA with SHA-1 — RSA with MD5
Encrypt session key for transmission with a message.	MUST support RSA encryption SHOULD support — RSAES-OAEP — Diffie-Hellman ephemeral-static mode
Encrypt message for transmission with a one-time session key.	MUST support AES-128 with CBC SHOULD support — AES-192 CBC and AES-256 CBC — Triple DES CBC

The following rules, in the following order, should be followed by a sending agent.

1. If the sending agent has a list of preferred decrypting capabilities from an intended recipient, it SHOULD choose the first (highest preference) capability on the list that it is capable of using.
2. If the sending agent has no such list of capabilities from an intended recipient but has received one or more messages from the recipient, then the outgoing message SHOULD use the same encryption algorithm as was used on the last signed and encrypted message received from that intended recipient.
3. If the sending agent has no knowledge about the decryption capabilities of the intended recipient and is willing to risk that the recipient may not be able to decrypt the message, then the sending agent SHOULD use triple DES.
4. If the sending agent has no knowledge about the decryption capabilities of the intended recipient and is not willing to risk that the recipient may not be able to decrypt the message, then the sending agent MUST use RC2/40.

If a message is to be sent to multiple recipients and a common encryption algorithm cannot be selected for all, then the sending agent will need to send two messages. However, in that case, it is important to note that the security of the message is made vulnerable by the transmission of one copy with lower security.

S/MIME Messages

S/MIME makes use of a number of new MIME content types. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort.

We examine each of these in turn after first looking at the general procedures for S/MIME message preparation.

SECURING A MIME ENTITY S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 5322 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. The MIME entity is prepared according to the normal rules for MIME message preparation. Then the MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce what is known as a PKCS object. A PKCS object is then treated as message content and wrapped in MIME (provided with appropriate MIME headers). This process should become clear as we look at specific objects and provide examples.

In all cases, the message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart.

The use of transfer encoding requires special attention. For most cases, the result of applying the security algorithm will be to produce an object that is partially or totally represented in arbitrary binary data. This will then be wrapped in an outer MIME message and transfer encoding can be applied at that point, typically base64. However, in the case of a multipart signed message (described in more detail later), the message content in one of the subparts is unchanged by the security process. Unless that content is 7 bit, it should be transfer encoded using base64 or quoted-printable so that there is no danger of altering the content to which the signature was applied.

We now look at each of the S/MIME content types.

ENVELOPEDDATA An application/pkcs7-mime subtype is used for one of four categories of S/MIME processing, each with a unique smime-type parameter. In all cases, the resulting entity, (referred to as an *object*) is represented in a form known as Basic Encoding Rules (BER), which is defined in ITU-T Recommendation X.209. The BER format consists of arbitrary octet strings and is therefore binary data. Such an object should be transfer encoded with base64 in the outer MIME message. We first look at envelopedData.

The steps for preparing an envelopedData MIME entity are:

1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or triple DES).
2. For each recipient, encrypt the session key with the recipient's public RSA key.
3. For each recipient, prepare a block known as `RecipientInfo` that contains an identifier of the recipient's public-key certificate,¹ an identifier of the algorithm used to encrypt the session key, and the encrypted session key.
4. Encrypt the message content with the session key.

¹This is an X.509 certificate, discussed later in this section.

The `RecipientInfo` blocks followed by the encrypted content constitute the `envelopedData`. This information is then encoded into base64. A sample message (excluding the RFC 5322 headers) is given below.

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-
    data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBgHyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTTrfvbnjT6jH7756tbB9H
f8HHGTTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

To recover the encrypted message, the recipient first strips off the base64 encoding. Then the recipient's private key is used to recover the session key. Finally, the message content is decrypted with the session key.

SIGNED DATA The `signedData` smime-type can be used with one or more signers. For clarity, we confine our description to the case of a single digital signature. The steps for preparing a `signedData` MIME entity are as follows.

1. Select a message digest algorithm (SHA or MD5).
2. Compute the message digest (hash function) of the content to be signed.
3. Encrypt the message digest with the signer's private key.
4. Prepare a block known as `SignerInfo` that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

The `signedData` entity consists of a series of blocks, including a message digest algorithm identifier, the message being signed, and `SignerInfo`. The `signedData` entity may also include a set of public-key certificates sufficient to constitute a chain from a recognized root or top-level certification authority to the signer. This information is then encoded into base64. A sample message (excluding the RFC 5322 headers) is the following.

```
Content-Type: application/pkcs7-mime; smime-type=signed-
    data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBgHyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

To recover the signed message and verify the signature, the recipient first strips off the base64 encoding. Then the signer's public key is used to decrypt the message digest. The recipient independently computes the message digest and compares it to the decrypted message digest to verify the signature.

CLEAR SIGNING Clear signing is achieved using the multipart content type with a signed subtype. As was mentioned, this signing process does not involve transforming the message to be signed, so that the message is sent “in the clear.” Thus, recipients with MIME capability but not S/MIME capability are able to read the incoming message.

A multipart/signed message has two parts. The first part can be any MIME type but must be prepared so that it will not be altered during transfer from source to destination. This means that if the first part is not 7 bit, then it needs to be encoded using base64 or quoted-printable. Then this part is processed in the same manner as signedData, but in this case an object with signedData format is created that has an empty message content field. This object is a detached signature. It is then transfer encoded using base64 to become the second part of the multipart/signed message. This second part has a MIME content type of application and a subtype of pkcs7-signature. Here is a sample message:

```
Content-Type: multipart/signed;
    protocol="application/pkcs7-signature";
    micalg=sha1; boundary=boundary42
--boundary42
Content-Type: text/plain
This is a clear-signed message.
--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
--boundary42--
```

The protocol parameter indicates that this is a two-part clear-signed entity. The micalg parameter indicates the type of message digest used. The receiver can verify the signature by taking the message digest of the first part and comparing this to the message digest recovered from the signature in the second part.

REGISTRATION REQUEST Typically, an application or user will apply to a certification authority for a public-key certificate. The application/pkcs10 S/MIME

entity is used to transfer a certification request. The certification request includes `certificationRequestInfo` block, followed by an identifier of the public-key encryption algorithm, followed by the signature of the `certificationRequestInfo` block, made using the sender's private key. The `certificationRequestInfo` block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the user's public key.

CERTIFICATES-ONLY MESSAGE A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request. The message is an `application/pkcs7-mime` type/subtype with an `smime-type` parameter of `degenerate`. The steps involved are the same as those for creating a `signedData` message, except that there is no message content and the `signerInfo` field is empty.

S/MIME Certificate Processing

S/MIME uses public-key certificates that conform to version 3 of X.509 (see Chapter 14). S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists. That is, the responsibility is local for maintaining the certificates needed to verify incoming signatures and to encrypt outgoing messages. On the other hand, the certificates are signed by certification authorities.

USER AGENT ROLE An S/MIME user has several key management functions to perform.

- **Key generation:** The user of some related administrative utility (e.g., one associated with LAN management) **MUST** be capable of generating separate Diffie–Hellman and DSS key pairs and **SHOULD** be capable of generating RSA key pairs. Each key pair **MUST** be generated from a good source of nondeterministic random input and be protected in a secure fashion. A user agent **SHOULD** generate RSA key pairs with a length in the range of 768 to 1024 bits and **MUST NOT** generate a length of less than 512 bits.
- **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.
- **Certificate storage and retrieval:** A user requires access to a local list of certificates in order to verify incoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users.

Enhanced Security Services

RFC 2634 defines four enhanced security services for S/MIME:

- **Signed receipts:** A signed receipt may be requested in a `SignedData` object. Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message. In essence, the recipient signs the entire original message plus the original (sender's) signature and appends the new signature to form a new S/MIME message.

- **Security labels:** A security label may be included in the authenticated attributes of a `SignedData` object. A security label is a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. The labels may be used for access control, by indicating which users are permitted access to an object. Other uses include priority (secret, confidential, restricted, and so on) or role based, describing which kind of people can see the information (e.g., patient's health-care team, medical billing agents).
- **Secure mailing lists:** When a user sends a message to multiple recipients, a certain amount of per-recipient processing is required, including the use of each recipient's public key. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message. The originator of a message need only send the message to the MLA with encryption performed using the MLA's public key.
- **Signing certificates:** This service is used to securely bind a sender's certificate to their signature through a signing certificate attribute.

19.5 PRETTY GOOD PRIVACY

An alternative email security protocol is Pretty Good Privacy (PGP), which has essentially the same functionality as S/MIME. PGP was created by Phil Zimmerman and implemented as a product first released in 1991. It was made available free of charge and became quite popular for personal use. The initial PGP protocol was proprietary and used some encryption algorithms with intellectual property restrictions. In 1996, version 5.x of PGP was defined in IETF RFC 1991, *PGP Message Exchange Formats*. Subsequently, OpenPGP was developed as a new standard protocol based on PGP version 5.x. OpenPGP is defined in RFC 4880 (*OpenPGP Message Format*, November 2007) and RFC 3156 (*MIME Security with OpenPGP*, August 2001).

There are two significant differences between S/MIME and OpenPGP:

- **Key Certification:** S/MIME uses X.509 certificates that are issued by Certificate Authorities (or local agencies that have been delegated authority by a CA to issue certificates). In OpenPGP, users generate their own OpenPGP public and private keys and then solicit signatures for their public keys from individuals or organizations to which they are known. Whereas X.509 certificates are trusted if there is a valid PKIX chain to a trusted root, an OpenPGP public key is trusted if it is signed by another OpenPGP public key that is trusted by the recipient. This is called the *Web-of-Trust*.
- **Key Distribution:** OpenPGP does not include the sender's public key with each message, so it is necessary for recipients of OpenPGP messages to separately obtain the sender's public key in order to verify the message. Many organizations post OpenPGP keys on TLS-protected websites: People who wish to verify digital signatures or send these organizations encrypted mail

need to manually download these keys and add them to their OpenPGP clients. Keys may also be registered with the OpenPGP public key servers, which are servers that maintain a database of PGP public keys organized by email address. Anyone may post a public key to the OpenPGP key servers, and that public key may contain any email address. There is no vetting of OpenPGP keys, so users must use the Web-of-Trust to decide whether to trust a given public key.

NIST 800-177 recommends the use of S/MIME rather than PGP because of the greater confidence in the CA system of verifying public keys.

Appendix P provides an overview of PGP.

19.6 DNSSEC

DNS Security Extensions (DNSSEC) are used by several protocols that provide email security. This section provides a brief overview of the Domain Name System (DNS) and then looks at DNSSEC.

Domain Name System

DNS is a directory lookup service that provides a mapping between the name of a host on the Internet and its numeric IP address. DNS is essential to the functioning of the Internet. The DNS is used by MUAs and MTAs to find the address of the next hop server for mail delivery. Sending MTAs query DNS for the Mail Exchange Resource Record (MX RR) of the recipient's domain (the right hand side of the "@" symbol) in order to find the receiving MTA to contact.

Four elements comprise the DNS:

- **Domain name space:** DNS uses a tree-structured name space to identify resources on the Internet.
- **DNS database:** Conceptually, each node and leaf in the name space tree structure names a set of information (e.g., IP address, name server for this domain name) that is contained in resource record. The collection of all RRs is organized into a distributed database.
- **Name servers:** These are server programs that hold information about a portion of the domain name tree structure and the associated RRs.
- **Resolvers:** These are programs that extract information from name servers in response to client requests. A typical client request is for an IP address corresponding to a given domain name.

THE DNS DATABASE DNS is based on a hierarchical database containing **resource records (RRs)** that include the name, IP address, and other information about hosts. The key features of the database are as follows:

- **Variable-depth hierarchy for names:** DNS allows essentially unlimited levels and uses the period (.) as the level delimiter in printed names, as described earlier.