Essay 17

Internet Privacy Enhanced Mail

Stephen T. Kent

Privacy Enhanced Mail (PEM) consists of extensions to existing message processing software plus a key management infrastructure. These combine to provide users with a facility in which message confidentiality, authenticity, and integrity can be effected. PEM is compatible with RFC 822 message processing conventions and is transparent to SMTP mail relays. PEM uses symmetric cryptography — for example, the Data Encryption Standard (DES) — to provide (optional) encryption of messages. Although the RFCs permit the use of either symmetric or asymmetric (public key) cryptography (for instance, the RSA cryptosystem) to distribute symmetric keys, the RFCs strongly recommend the use of asymmetric cryptography for this purpose and to generate and validate digital signatures for messages and certificates. Public key management in PEM is based on the use of certificates as defined by the CCITT Directory Authentication Framework [CCIT88c]. A public key certification hierarchy for PEM is being established by the Internet Society. This certification hierarchy supports universal authentication of PEM users, under various policies, without the need for prior bilateral agreements among users or organizations with which the users may be affiliated.

The primary focus of the effort to develop and deploy Privacy Enhanced Mail (PEM) is the provision of security services for e-mail users in the Internet community.[1] This effort began in 1985 as an activity [LINN86] of the Privacy and Security Research Group (PSRG)[2] or the Internet Re-

[1] The Internet e-mail community is interpreted here to include those users who use the protocols defined by RFC 821 and RCC 822 for e-mail.

[2] The PSRG was formed in 1985 and is one of several groups that pursue various research topics in the context of the Internet. Other groups have been constituted to explore topics such as end-to-end protocols, multimedia teleconferencing, information location services, and so on.

search Task Force, under the auspices of the Internet Architecture Board (IAB).[3] The effort has yielded a series of Requests for Comment (RFCs)[4] of which the most recent set, RFCs 1421 to 1424 [LINN93, KENT93, BALE93, KALA93], are Proposed Internet Standards. This essay describes the version of PEM defined by those RFCs. Ongoing work to integrate PEM with MIME [BORE92] is not detailed since, at the time at which this essay was prepared, that work was not yet stable.

Overview

PEM provides several security services for e-mail users: confidentiality, data origin authentication, and connectionless integrity, as defined by ISO [ISO89]. If appropriate algorithms are used, PEM also provides support for nonrepudiation with proof of origin. These services are bundled into two groups: All messages processed through PEM incorporate the authenticity, integrity, and nonrepudiation support facilities, whereas confidentiality is an optional security service.

The integrity and authenticity services ensure a message recipient that a message was sent by the indicated originator and that the message has not been modified en route. The nonrepudiation mechanisms allow a message to be forwarded to a third party, who can verify the identity of the originator (not just the identity of the forwarder) and verify that the message has not been altered, even by the original recipient. The optional confidentiality service ensures a message originator that the message text will be disclosed (decipherable) only to the designated recipients.

As noted above, PEM is intended for use with existing e-mail systems, primarily Internet e-mail as defined by RFC 822. Figure 1 illustrates how PEM can be integrated into existing mail system architectures. A variety of environments are illustrated here, including message preparation on a multiuser host incorporating a message transfer agent (MTA) and relay through an intermediate MTA.[5] In the figure, messages are retrieved from a mailbox on a computer that is separate from the recipient's worksta-

[3] The IAB oversees the development of the Internet architecture and the execution of the Internet standards process, under the auspices of the Internet Society.

[4] RFCs are the official, archival publications of the IAB. All protocol standards are published as RFCs, but not all RFCs are protocol standards. For example, some RFCs are merely informational, others describe experimental protocols, and still others constitute policy statements. RFCs are available on line from various sites and in hard-copy form from SRI International and the InterNIC.

[5] The concepts of UAs and MTAs are taken from X.400 [CCIT88a] but apply equally well to messaging in the TCP/IP environment. In the TCP/IP protocol suite, MTAs are represented by SMTP (defined in RFC 821) processes, which route and relay mail traffic. UAs are represented by processes that implement RFC 822 message processing.

tion, for example, using the Post Office Protocol (POP) [ROSE91] (or P7 in an X.400 environment). To maximize compatibility with such systems, PEM is designed to be transparent to mail transfer systems, so that the existing transport infrastructure can be used for PEM. PEM also is designed to be minimally intrusive to mail system user agents, although transparency here seems to entail trade-offs in quality of the user interface.

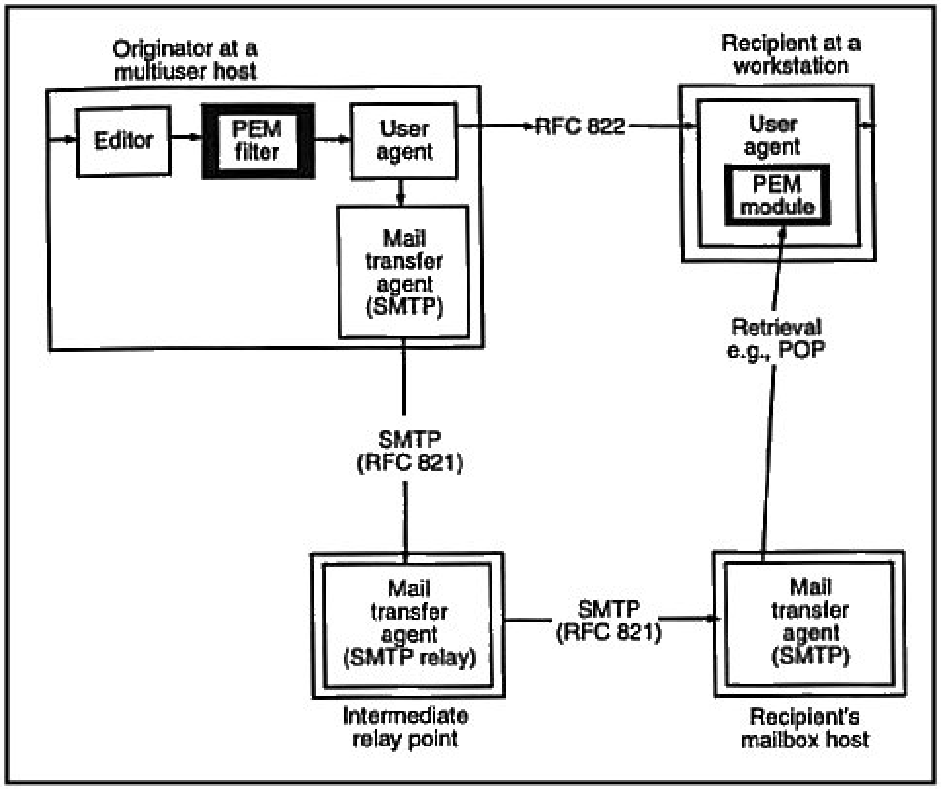

Figure 1. PEM environment.

One can implement PEM as a filter applied to a file created using an editor but prior to submission to a mail system user agent. However, this approach may provide a poor user interface unless a substantial amount of mechanism from the user agent is replicated in the PEM filter or the editor (for example, to deal with acquisition of recipient addresses). In contrast, if PEM processing is integrated into a user agent, the user agent must provide new user interface facilities to allow selection of security services, but the resulting interface can be especially "user friendly." Both approaches to PEM implementation are illustrated in Figure 1.

Acronyms used in this essay

ASCII: American Standard Code for Information Interchange
ASN: Abstract Syntax Notation
CA: Certification Authority
CBC: Cipher block chaining
CCITT: International Telegraph and Telephone Consultative Committee, now called the International Telecommunications Union — Telecommunications Standardization Sector (ITU-T)
CRL: Certificate revocation list
DES: Data Encryption Standard
DSA: Directory Services Agent (Note: In cryptography, DSA is the Digital Signature Algorithm.)
IAB: Internet Architecture Board
IPRA: Internet PCA Registration Authority
ISO: International Standards Organization
MIC: Message integrity code
MTA: Message transfer agent
OSI: Open Systems Interconnection
PCA: Policy Certification Authority
PSRG: Privacy and Security Research Group
PEM: Privacy Enhanced Mail
POP: Post Office Protocol
RFC: Request for Comments
RSA: Rivest, Shamir, and Adleman
SMTP: Simple Message Transfer Protocol
TCP/IP: Transmission Control Protocol/Internet Protocol
UA: User agent

In providing these security services, PEM uses a variety of cryptographic algorithms. These algorithms provide for message integrity, message encryption, digital signatures, and distribution of keys used to encipher messages. If public key algorithms are used in this latter context, then two additional algorithms must be specified: one for certificate hashing and one for certificate signatures. The base PEM standards do not require the use of specific algorithms for any of these purposes, but rather provide facilities to identify which algorithms are used on a per-message and per-recipient basis. A separate standard within the PEM series (initially, RFC 1115) provides specifications for the use of specific algorithms with PEM.

PEM is oriented primarily toward use in the Internet e-mail environment, as characterized by the use of two Internet standards: RFC 822 [CROC82] and SMTP [POST82]. The former defines the syntax and (header) semantics for messages, while the latter defines the protocol for transport of messages. Although designed expressly for use with these protocols, PEM can be used in a wider range of messaging environments; for example, the NIST Open Implementors Workshop has defined an X.400 body part to carry PEM messages. PEM-processed messages intentionally employ a 6-bit encoding that utilizes a subset of printable characters to maximize the likelihood that such messages can transit the mail gateways (many of which do not provide transparent forwarding of message contents) that link Internet e-mail to other e-mail systems — BITNET, UUNET, and so on.

In addition to the specification of message processing facilities, the PEM standards provide for a public key certification infrastructure. Although PEM allows for the use of either secret key (symmetric) or public key (asymmetric) cryptoalgorithms for key distribution, the standards encourage the use of public key cryptography because of its ability to support a very large, distributed user community. The specific approach to public key cryptography adopted for PEM is based on the use of certificates, as defined in CCITT Recommendation X.509.

The PEM standards establish a specific framework for a public key certification system for several reasons. Although PEM makes use of X.509 certificates, this CCITT recommendation does not provide a semantic context in which to interpret certificates. X.509 embodies a degree of generality that, if fully exploited, could result in rather complex certification relationships. The PEM certification system imposes conventions that make certification relationships straightforward and allow users to readily interpret each certificate associated with other PEM users. Another advantage of establishing this certification framework is that it can be used with other security protocols, for example, X.500 and X.400. The same certificates used for PEM could be employed with these other applications in support of security services.

PEM message submission: Message processing

The overall flow of data for PEM message submission processing is illustrated in Figure 2. There are two sources of data input to PEM for message submission: message header and message text. The message header information will be carried in the RFC 822 header of the final, processed PEM message. This data largely bypasses PEM processing, with the possible exception that the Subject field, if deemed sensitive, might be omitted or a benign Subject might be substituted (for example, "Encrypted Message"). This header is separated from the remainder of the data by a blank line, following the usual convention, and then an explicit boundary marker is inserted to identify this as a PEM message.

Some of the header data, namely recipient addresses, serve as input to PEM processing to control the (optional) message encryption function, which generates PEM encapsulated header data. Following this PEM message header data is another blank line. The message text itself, possibly augmented by header fields replicated so that they can be afforded protection, becomes the encapsulated text of the PEM-processed message, following the (second) blank line. Finally, a complementary PEM boundary message completes the PEM message.

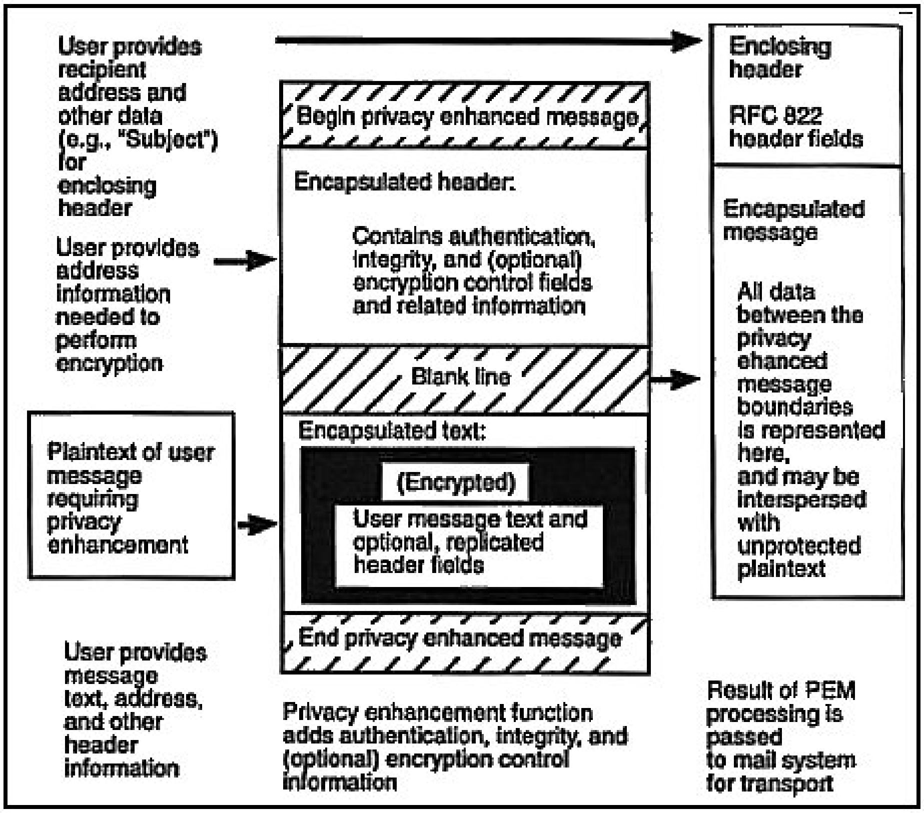
User provides recipient address and other data (e.g., "Subject") for enclosing header
Enclosing header
RFC 822 header fields
Begin privacy enhanced message
Encapsulated header:
Contains authentication, integrity, and (optional) encryption control fields and related information
User provides address information needed to perform encryption
Encapsulated message
All data between the privacy ehanced message boundaries is represented here, and may be interspersed with unprotected plaintext
Blank line
Encapsulated text:
Plaintext of user message requiring privacy enhancement
(Encrypted)
User message text and optional, replicated header fields
End privacy enhanced message
User provides message text, address, and other header information
Privacy enhancement function adds authentication, integrity, and (optional) encryption control information
Result of PEM processing is passed to mail system for transport

Figure 2. PEM message submission overview.

PEM message processing involves three major transformation steps: SMTP canonicalization, computation of the message integrity code, and optional message encryption followed by optional 6-bit encoding. These steps are illustrated in Figure 3.

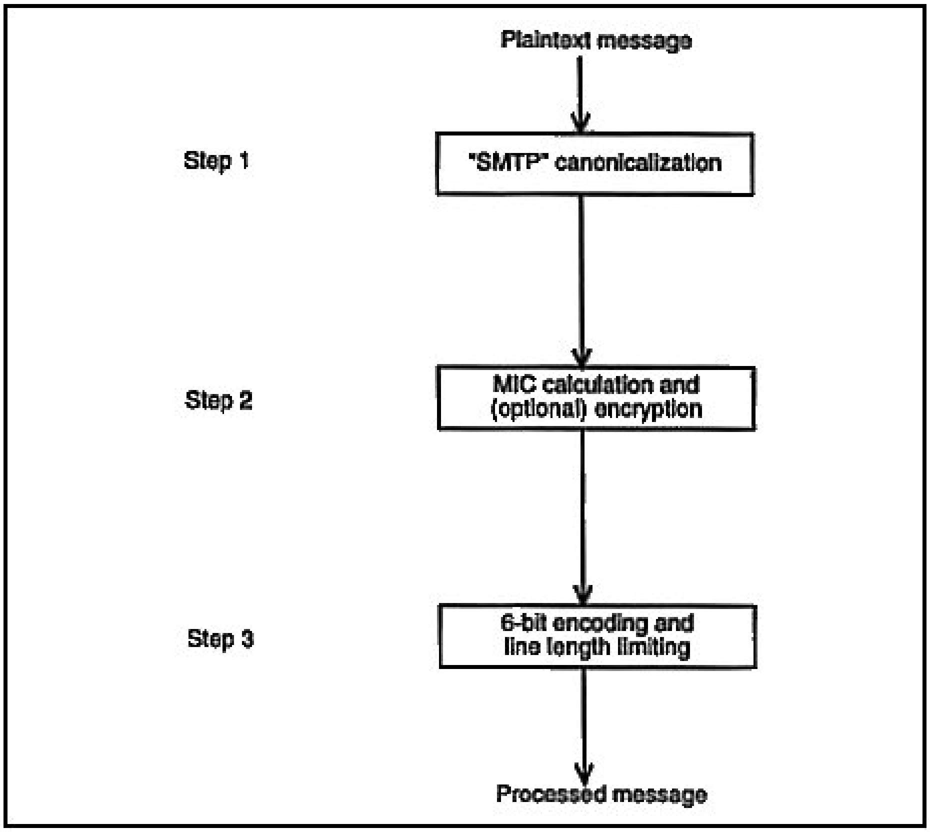
Plaintext message
Step 1
"SMTP" canonicalization
Step 2
MIC calculation and (optional) encryption
Step 3
6-bit encoding and line length limiting
Processed message

Figure 3. PEM processing steps.

The first step uses the canonicalization specified by SMTP to ensure a uniform presentation syntax among a heterogeneous collection of computer systems. A shortcoming of this particular choice of processing — as opposed to a more general presentation syntax such as the ASN.1 concrete encoding used in X.400 [CCIT88a] — is that it restricts the input to 7-bit ASCII. However, this specific canonicalization was selected because the primary application environment, that is, Internet e-mail, imposed the same restrictions.

The second step begins with the calculation of the message integrity code (MIC). PEM allows this function to be as simple as a DES message authentication code if the message is directed to only a single recipient. However, most messages will be addressed to multiple recipients, and in such circumstances a one-way hash function is required to prevent any one recipient from spoofing others. Moreover, support for nonrepudiation with proof of origin is provided only if a one-way hash is used. The MIC is calculated on the canonicalized version of the message to permit uniform verification in the heterogeneous environment alluded to above. The specific algorithm used for MIC computation is specified in the MIC-Info field of the PEM header (Figure 4).

The second step also provides message encryption (if selected by the originator). If the message is to be encrypted, any padding required by the message encryption algorithm is first applied. A message key, to be used exclusively to encrypt this message, is generated. The (symmetric) encryption algorithm employed is specified in the DEK-Info field in the PEM header (Figure 4), along with any parameters required by the algorithm (for example, an initialization vector). The canonical (padded as required) message text is then encrypted using the per-message key. All of these actions are performed only if the message is of type ENCRYPTED.

The third (final) processing step renders an ENCRYPTED or MIC-ONLY message into a printable form suitable for transmission via SMTP and across a variety of messaging system boundaries. This encoding step transforms the (optionally encrypted) message text into a restricted 6-bit alphabet (plus line length constraints that make the encoding compatible with SMTP canonicalization). If the message has been encrypted, this encoding serves to transform the resulting 8-bit ciphertext into a form that can be transmitted using SMTP and other message transfer protocols (which require 7-bit ASCII). MIC-CLEAR messages are not subject to any portion of the third processing step. A MIC-CLEAR message is a signed, but not encrypted, message that is not encoded, specifically so that it can be sent to a mixed set of recipients, some of whom use PEM and some who do not.

Even if the message has not been encrypted, this encoding ensures, with high probability, that the canonicalized version of the message (produced in step 1) will not be altered (benignly) in transit, for example, while passing through a mail gateway. A change in as little as one bit of the text would cause the MIC check to fail at a destination, hence the need to ensure that the transformed message text can be transmitted without modification. Because MIC-CLEAR messages are not encoded, they are susceptible to (benign) gateway manipulation and thus run an increased risk of failing the MIC check at recipients who perform PEM processing. The provision of the MIC-CLEAR message type in PEM thus represents an explicit trade-off between immunity to (benign) transport manipulation and flexibility in sending mail to mixed user communities.

PEM message submission: Header construction

After the processing steps described above have been accomplished, the PEM message header is constructed. The precise steps followed at this point depend on whether secret key or public key cryptography is used to distribute keys. Since the public key approach is expected to be most

```
 To:      Linn@dec.com
 From:    Kent@bbn.com
 Subject: Encrypted PEM Message
 -----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: 4,ENCRYPTED
Content-Domain: RFC822
DEK-Info: DES-CBC,BFF968AA74691AC1
Originator-Certificate:
 MIIBlTCCAScCAWUwDQYJKoZIhvcNAQECBQAwUTELMAkGA1UEBhMCVVMxIDAeBgNV
 BAoTF1JTQSBEYXRhIFNlY3VyaXR5LCBJbmMuMQ8wDQYDVQQLEwZCZXRhIDExDzAN
 BgNVBAsTBk5PVEFSWTAeFw05MTA5MDQxODM4MTdaFw05MzA5MDMxODM4MTZaMEUx
 CzAJBgNVBAYTAlVTMSAwHgYDVQQKExdSU0EgRGF0YSBTZWN1cml0eSwgSW5jLjEU
 MBIGA1UEAxMLVGVzdCBVc2VyIDEwWTAKBgRVCAEBAgICAANLADBIAkEAwHZHl7i+
 yJcqDtjJCowzTdBJrdAiLAnSC+CnnjOJELyuQiBgkGrgIh3j8/x0fM+YrsyF1u3F
 LZPVtzlndhYFJQIDAQABMA0GCSqGSIb3DQEBAgUAA1kACKr0PqphJYw1j+YPtcIq
 iWlFPuN5jJ79Khfg7ASFxskYkEMjRNZV/HZDZQEhtVaU7Jxfzs2wfX5byMp2X3U/
 5XUXGx7qusDgHQGs7Jk9W8CW1fuSWUgN4w==
Key-Info: RSA,
 I3rRIGXUGWAF8js5wCzRTkdhO34PTHdRZY9Tuvm03M+NM7fx6qc5udixps2Lng0+
 wGrtiUm/ovtKdinz6ZQ/aQ==
Issuer-Certificate:
 MIIB3DCCAUgCAQowDQYJKoZIhvcNAQECBQAwTzELMAkGA1UEBhMCVVMxIDAeBgNV
 BAoTF1JTQSBEYXRhIFNlY3VyaXR5LCBJbmMuMQ8wDQYDVQQLEwZCZXRhIDExDTAL
 BgNVBAsTBFRMQ0EwHhcNOTEwOTAxMDgwMDAwWhcNOTIwOTAxMDc1OTU5WjBRMQsw
 CQYDVQQGEwJVUzEgMB4GA1UEChMXUlNBIERhdGEgU2VjdXJpdHksIEluYy4xDzAN
 BgNVBAsTBkJldGEgMTEPMA0GA1UECxMGTk9UQVJZMHAwCgYEVQgBAQICArwDYgAw
 XwJYCsnp6lQCxYykNlODwutF/jMJ3kL+3PjYyHOwk+/9rLg6X65B/LD4bJHtO5XW
 cqAz/7R7XhjYCm0PcqbdzoACZtIlETrKrcJiDYoP+DkZ8k1gCk7hQHpbIwIDAQAB
 MA0GCSqGSIb3DQEBAgUAA38AAICPv4f9Gx/tY4+p+4DB7MV+tKZnvBoy8zgoMGOx
 dD2jMZ/3HsyWKWgSF0eH/AJB3qr9zosG47pyMnTf3aSy2nBO7CMxpUWRBcXUpE+x
 EREZd9++32ofGBIXaialnOgVUn0OzSYgugiQ077nJLDUj0hQehCizEs5wUJ35a5h
MIC-Info: RSA-MD5,RSA,
 UdFJR8u/TIGhfH65ieewe2lOW4tooa3vZCvVNGBZirf/7nrgzWDABz8w9NsXSexv
 AjRFbHoNPzBuxwmOAFeA0HJszL4yBvhG
Recipient-ID-Asymmetric:
 MFExCzAJBgNVBAYTAlVTMSAwHgYDVQQKExdSU0EgRGF0YSBTZWN1cml0eSwgSW5j
 LjEPMA0GA1UECxMGQmV0YSAxMQ8wDQYDVQQLEwZOT1RBUlk=,
 66
Key-Info: RSA,
 O6BS1ww9CTyHPtS3bMLD+L0hejdvX6Qv1HK2ds2sQPEaXhX8EhvVphHYTjwekdWv
 7x0Z3Jx2vTAhOYHMcqqCjA==

qeWlj/YJ2Uf5ng9yznPbtD0mYloSwIuV9FRYx+gzY+8iXd/NQrXHfi6/MhPfPF3d
jIqCJAxvld2xgqQimUzoS1a4r7kQQ5c/Iua4LqKeq3ciFzEv/MbZhA==
-----END PRIVACY-ENHANCED MESSAGE-----
```

Figure 4. Sample PEM message.

widely used and its use is strongly encouraged by the specifications, this essay focuses on that technique. Figure 4 displays a sample PEM message, with an RFC 822 header, and the following text is keyed to this example. The sample message makes use of version 4 of the protocol and is encrypted, as indicated by the Proc-Type field.

To provide data origin authentication and message integrity, and to support nonrepudiation with proof of origin, the MIC computed above in step 2 is padded and then encrypted using the private component of the originator's public key pair. This effects a digital signature on the message, which can be verified by any user employing the originator's public component. If the message is encrypted, this signature value is encrypted using the secret, per-message key, which was used to encrypt the message text itself. This last encryption step is used to protect against the disclosure of a (trivial) repetitive message content that could be discerned by observation of the signed hash value. The resulting value is 6-bit encoded and included in the MIC-Info field (Figure 4), along with the identifiers of the MIC algorithm and the digital signature algorithm. In this example, the MD5 hash function is used as the MIC algorithm, and the RSA algorithm is used as the digital signature algorithm.

If the message is encrypted, the algorithm used to encrypt the message and any parameters required by the (message) encryption algorithm are specified in the DEK-Info field, which appears once per message. In this example, the CBC mode of DES is used as the encryption algorithm, and the initialization vector used is represented as the second component of this field.

To provide confidentiality, one copy of the message key is encrypted using the public component of the public key pair for each recipient. In this way, each copy of the message key is protected in a fashion that makes it decipherable by exactly one recipient. The result of this encryption is placed in the Key-Info field, following an identifier for the public key algorithm used to encrypt the message key. Each Key-Info field is preceded by a Recipient-ID-Asymmetric field that identifies the recipient (by the X.500-distinguished name[6] of his certificate issuer and certificate serial number). Thus each pair of these PEM header fields together provides the information required for a recipient to decrypt a message. Here RSA is used as the public key encryption algorithm and is so identified in the Key-Info field.

The originator may, at his discretion, provide his certificate (using the Originator-Certificate field) and the certificate of the entity that issued his certificate, and so on (using multiple Issuer-Certificate fields as required).

[6] Each entry in an X.500 directory server has a unique name which identifies that entry, called the "distinguished name" of the entry. That name is composed of selected attributes from superior entries in the directory tree, all the way back to the root.

The inclusion of these certificates is intended to facilitate the certificate validation process by recipients, but is not required. This example illustrates the inclusion of the originator's certificate and one issuer certificate.

PEM message delivery processing

On receipt of a PEM-protected message, a recipient first scans the PEM header to identify the version of PEM that was used and the form of PEM processing that has been applied: ENCRYPTED, MIC-ONLY, or MIC-CLEAR. The message type determines the processing steps performed by the recipient.

If the message is ENCRYPTED or MIC-ONLY, the first step is the inversion of the encoding process applied by the originator, converting the 6-bit encoding back into the ciphertext or canonical plaintext form. If the message is ENCRYPTED, the recipient scans the PEM header to locate the Recipient-ID-Asymmetric field for this recipient. The recipient uses the private component of his public key pair to decrypt the associated Key-Info field, yielding the message key and extracting parameters associated with the message encryption algorithm. The recipient now uses the message key to decrypt the message text. After decryption, the message is now at the same processing status as a MIC-ONLY or MIC-CLEAR message, and the following text applies to those message types as well. (In fact, a MIC-CLEAR message requires a processing step unique to that message type. The step is the recanonicalization of the message insofar as lines are delimited by a carriage return and a line feed, versus a local representation of delimited lines.)

The recipient now processes the MIC-Info field, using the MIC algorithm and signature algorithm specified in this field. The recipient needs to acquire the public component of the originator to check the signature. In principle, this requires validating a chain of certificates that terminates with the certificate of the originator, although caching of certificates by user agents is expected to short-circuit this process in most instances. Using this public component, the recipient decrypts the signature, revealing the MIC value. The recipient computes the MIC on the canonical form of the message and compares the result with the decrypted value. If they match, the integrity and data origin authenticity of the message are verified. This verification step applies to all three message forms.

Finally, after verifying message integrity, the canonical form of the message is translated into the local representation apropos for the recipient's system and is displayed for the user. The recipient also should be informed of the cryptographically authenticated originator identity through some out-of-band means, that is, separate from the "From" field contained within the message. For example, in a graphical user interface system this notification could be effected using a window separate from that used to display the message text. Errors encountered in attempting to validate message integrity or decrypt the message may result in informative messages or may preclude display of the message for the recipient, depending on the severity of the error and on local security policy.

The PEM certification system

The PEM specifications recommend use of public key cryptography for message integrity and authentication, and key distribution for message encryption keys. As noted earlier, PEM makes use of public key certificates that conform to the CCITT X.509 recommendations. This recommendation defines an authentication framework in which certificates play a central role, which is quite general and places very few constraints on the resulting certification system. PEM "profiles" this framework, imposing conventions that result in a concrete realization of a certification system that is a conformant subset of that envisioned in X.509.

X.509 certificates. A (public key) certificate is a data structure used to securely bind a public key to a name and to specify who vouches for the binding. The structure as a whole is digitally signed in roughly the same fashion as a PEM message is signed (for example, the canonicalization rules are different). What follows is the certificate format specified by X.509 using ASN.1 notation to define the top-level fields in the structure:

```
Certificate ::=          SIGNED SEQUENCE{
version [0]                    Version DEFAULT v1988,
serialNumber                         CertificateSerialNumber,
signature                      AlgorithmIdentifier,
issuer                       Name,
validity                             Validity,
subject                              Name,
subjectPublicKeyInfo           SubjectPublicKeyInfo}
```

The *version* field is used to differentiate among successive versions of the certificate format. The *serialNumber* field uniquely identifies this certificate among those issued by the same entity. The *signature* field specifies the digital signature algorithm used to sign this certificate. The *issuer* field specifies the directory (distinguished) name of the entity that vouches for the binding between the *subject* directory (distinguished) name and the public key contained in the certificate. The *validity* field specifies the start and end times and dates that delimit the interval over which the certificate is valid. The key alluded to above is contained, along with an identifier to specify the algorithm and any parameters required by the algorithm, in the *subjectPublicKeyInfo* field.

As noted above, the *signature* field specifies the algorithms and any parameters required to verify the digital signature applied to the certificate. Typically, both a one-way hash algorithm and a public key signature algorithm will be specified. This signature is applied by the issuer (using his private component) and appended after these other certificate fields. Appended to the certificate is a data structure that reproduces the algorithm identifiers and parameters needed to verify this signature (using the public component of the issuer). Note that the algorithm used to sign the certificate may be different from the algorithm with which the subject's public key is employed.

One validates a certificate by verifying the signature applied by the issuer of the certificate. Specifically, one computes the one-way hash (specified in the *signature* field) over the certificate, uses the public component of the issuer to decrypt the value in the appended signature, and compares the two resulting values. The issues of how one acquires the public component of the issuer and how one determines whether to trust the issuer to vouch for this binding are the subject of the next section.

The Internet certification hierarchy. As noted above, the act of validating a certificate requires the user to possess the public key of the issuer of the certificate. This issuer also will have a certificate, and thus the process of certificate validation is recursive and implicitly defines a (directed) certification graph. However, the process must conclude at some point, implying that the user holds a public key that he obtained through some out-of-band channel. In X.509 the certification graph is largely unconstrained, and thus might be an arbitrary graph, for example, including loops.

Certification authorities. X.509 defines a Certification Authority (CA) as "an authority trusted by one or more users to create and assign certificates." X.509 also imposes no constraints on the relationship between a certificate issuer, such as a Certification Authority, and a subject — for example, with regard to distinguished names. Different CAs are expected to issue certificates under different policies, for example, varying degrees of assurance in vouching for name-public key bindings. However, X.509 makes no provisions for users to learn what policies each CA applies in issuing certificates. This makes it hard for a user to assign semantics to the bindings implied by certification.

The Internet community is adopting a certification graph that takes the form of a (singly) rooted tree, illustrated in Figure 5. The root of this tree is designated the Internet PCA Registration Authority (IPRA) and will be operated under the auspices of the Internet Society. The IPRA provides a reference point from which all certificates in the Internet certification hierarchy can be validated. The IPRA establishes a common policy that applies to all certificates issued under this hierarchy. The IPRA issues certificates to a second tier of entities designated Policy Certification Authorities (PCAs), which, in turn, issue certificates to CAs. CAs issue certificates to (subordinate) CAs or directly to users (including mailing lists).

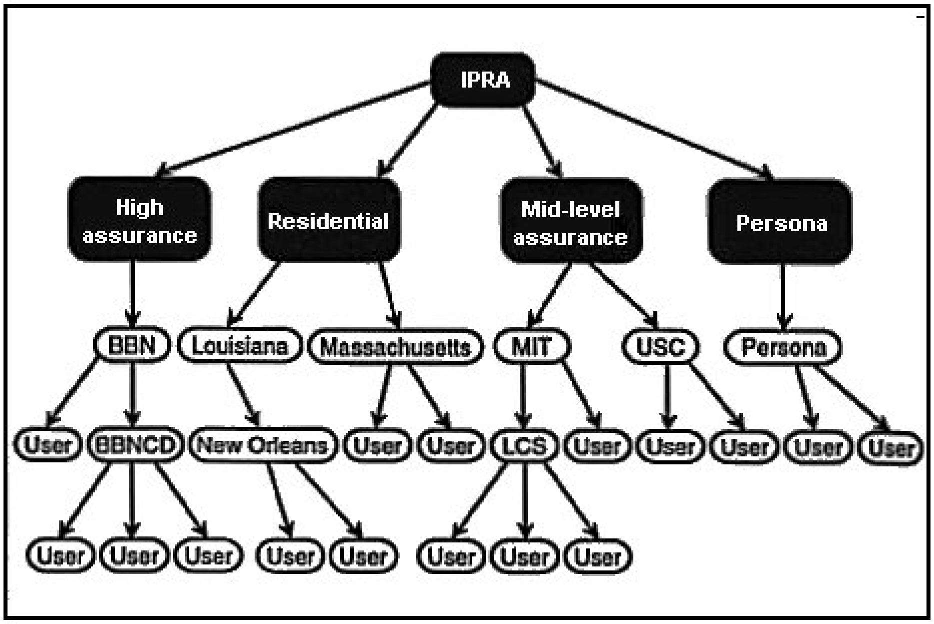

Figure 5. Proposed Internet certification hierarchy.

Typically, a CA will be certified by one PCA, and Figure 5 illustrates this common case. However, it is permissible for a single CA to be certified under multiple PCAs. In the latter case, the implication is that a single administrative entity is prepared to issue certificates under multiple disjoint policies. For each PCA under which a CA is certified, the CA certificate signed by the PCA must use a different public component. This ensures that the certificates issued by the CA under each policy are readily identifiable because each is signed using different private components. For example, in Figure 5, MIT is certified under both the mid-level and high assurance PCAs, and thus is capable of issuing certificates to faculty, staff, and students, based on either of the policies imposed by these PCAs.

In addition to the organizational CAs shown for MIT, BBN, and USC, residential CAs also are illustrated in Figure 5. These have been identified in more detail, using abbreviated forms of the distinguished name attributes for the geographic entities that they represent.[7] For example, two residential CAs are illustrated, one for Louisiana and another for Massachusetts, both within the United States. Below the Louisiana CA is a subordinate CA for the city (locality) of New Orleans.

IPRA common policy. The IPRA common policy is intended to encompass a minimum set of essential requirements that apply to all PCAs, CAs, and UAs. For example, this policy requires that each PCA file its statement of policy in accordance with a format that is also part of this policy. Each PCA policy must not contravene the IPRA common policy, but rather may specify additional policy aspects not addressed by the common policy. For example, a PCA policy statement will characterize the procedures used to authenticate CAs and users certified under this policy, plus any security requirements imposed on CAs for certificate management.

Each PCA policy statement must specify CRL management requirements, for example, upper and lower bounds on the frequency with which CAs must issue CRLs, provisions for archiving CRLs, and so on. The policy statement also must describe security measures used by the PCA in its management of certificates and to ensure the privacy of data held by the PCA in performing this task.

The IPRA policy also requires that all PCAs and CAs issue CRLs, although the policy does not constrain the frequency of issue of these CRLs. The IPRA also will establish and maintain a distributed database providing access to current CRLs for all PCAs and CAs within the community, at least until X.500 directory servers are sufficiently widely available to provide equivalent service.

The IPRA policy also requires PCAs and CAs to issue certificates in a fashion that ensures the uniqueness of distinguished names. To support PCAs in meeting this requirement, the IPRA will establish a registry of CA distinguished names, accessible by PCAs, which must be consulted before a PCA certifies a CA. An analogous but larger database will be established by the IPRA to record residential user distinguished names, to permit certification of such users by multiple PCAs. One approach to providing both of these databases involves having the IPRA operate an X.500 DSA, connected to other (extant) Internet DSAs. This would ensure that distinguished names associated with users and organizations registered in the existing X.500 system will not be infringed upon.

[7] The organizational CAs shown in Figure 5 do not contain their distinguished names due to space limitations. For example, the BBN CA might have a name of the form C=US, S=MA, O=BBN, where “O” is an abbreviation for the attribute “organization.”

A critical aspect of the IPRA policy deals with UA processing of certificates, rather than PCA or CA issuance of certificates. The fundamental requirement is distinguished name subordination. Whenever a certificate is validated for use in PEM, it must have the characteristic that the subject distinguished name in the certificate is subordinate to the issuer distinguished name in that certificate, unless the certificate in question is issued by the IPRA or by a PCA. This rule provides the user with a "natural" certification path that can be inferred by examination of the final certificate in the path, plus display of the distinguished name of the PCA under whose policy the certificate was issued.

The following example illustrates this rule. The BBN CA certificate might contain the following subject distinguished name: C=US, S=MA, O=BBN. This name would appear as the issuer name in all certificates issued by this CA. A valid distinguished name that could appear as the subject in a certificate issued by this CA would be C=US, S=MA, O=BBN, CN=Steve Kent (where CN is an abbreviation for the "common name" attribute). However, BBN would not be allowed to issue a certificate in which the subject name was of the form C=US, S=MA, O=MITRE, CN=Richard Parker. The subject name would not be subordinate to the BBN CA name as issuer. However, there is no subordination restriction on the relationship between a PCA and the CAs it certifies. Thus the BBN CA can be certified by a PCA with any distinguished name.

This constraint prohibits "cross-certification." When two CAs issue certificates in which each is a subject of the other CA, the result is termed "cross-certification." Cross-certification would short-circuit the path back to the PCA layer of the hierarchy, and thus prevent a user from ascertaining the policy under which a certificate was issued. Hence UAs must reject certification paths that entail cross-certification. As noted above, these requirements on certification paths are imposed on PEM UAs rather than on CAs. The reason for this is that CAs may sign certificates for use with other applications where cross-certification might be appropriate; hence the enforcement requirement is levied on PEM UAs.

Sample PCA policies. Although none is yet in place at the time of writing, several PCA policies have been proposed and are being refined. One such policy would serve businesses or other organizations that require a high degree of security from their use of PEM: a "high-assurance" PCA. This PCA would execute a legal agreement with each CA and require high-quality credentials to authenticate the CA. The PCA would require that the CA grant certificates to its users (for example, employees) using the same level of authentication that it would use in issuing ID cards. The CA would be required to issue CRLs at least monthly and not more often than weekly.

There would also be a requirement that the CA use highly secure technology, approved by the PCA, to generate and protect the CA's component pair and to sign all certificates issued by the CA. The PCA would use the same technology in generating its own component pair and in signing CA certificates. The PCA would promise to protect the privacy of all information provided by the CA during registration. This level of service is expected to require that the CA pay a registration fee to the PCA.

Another candidate PCA policy that has been put forth might be termed that of a "mid-level assurance" PCA. Here the validation of CA credentials would be less stringent, for example, written registration using a company letterhead might suffice. The CA would execute a very simple agreement, which would require a "good faith effort" to authenticate users. There would be no requirement to issue CRLs with any specific periodicity. Here each CA would be free to use any technology it deems appropriate to generate the CA component pair and to sign certificates. However, the PCA itself expects to use strong security technology to generate and protect its own component pair. This PCA envisions no charge to certify CAs, but would level a charge if a CA certificate had to be placed on the PCA's CRL.

A third PCA is envisioned to support residential users, that is, users not claiming affiliation with any organization. Such users could be registered using distinguished names based on geographic attributes, for example, country, state, locality, and street address. (In the US, a nine-digit ZIP code might be used in lieu of locality and street address data.) The user would be required to submit a notarized registration form as proof of his identity claim. In this context, the PCA is expected to operate "virtual" CAs representing geographic areas in advance of civil authorities offering this service. The PCA, through its virtual CA, would issue CRLs biweekly. User registration under this PCA is expected to entail a fee.

Finally, in support of personal privacy, a PCA has been proposed which would issue certificates that do *not* purport to express real user identities. These "persona" certificates will allow anonymous use of PEM while providing continuity of authenticity. Thus, even though one might not know the true identity of the holder of a persona certificate, one could determine if a series of messages originated under that identity were all from the same user (assuming the persona user does not share his private component. A PCA supporting persona users would ensure that all certificates issued under it are globally unique. This requirement is easily enforced by establishing a persona CA under the PCA and following the name subordination rules cited earlier. These certificates should not be confused with certificates that do purport to convey true identities, since there is no overlap in the name space and because a persona PCA must publish a policy statement declaring the nature of certificates issued by the CA(s) under it. A candidate PCA has proposed to issue persona certificates without charge, although it would charge a fee to place one of these certificates on the CRL managed by the PCA.

Conclusions

PEM represents a major effort to provide security for an application that touches a vast number of users within the Internet and beyond. PEM has been designed to accommodate gradual deployment in the Internet, both because of the backward compatibility with existing message transfer services and through provision of features such as MIC-CLEAR processing. The ultimate success of PEM will depend not only on the widespread availability of implementations for the range of hardware and software platforms used throughout the Internet, but also on successful establishment of the certification hierarchy that underlies asymmetric key management for PEM.

PEM was envisioned not as a long-term goal technology for secure messaging, but as an interim step before widespread availability of secure OSI messaging (and directory) services. However, depending on the viability of X.400 in the marketplace, PEM may become a long-term secure messaging technology rather than an interim step. In either case, PEM (or a successor) has the opportunity to become a crucial component in the evolution of the Internet as it paves the way for various mail-based applications that would not be possible without the underlying security services provided by PEM.