# Privacy-Enhanced Electronic Mail

*Matt Bishop*[1]

Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

*ABSTRACT*

The security of electronic mail sent through the internet may be described in exactly three words: there is none. The Privacy and Security Research Group has recommended implementing mechanisms designed to provide security enhancements. The first set of mechanisms provides a protocol to provide privacy, integrity, and authentication for electronic mail; the second provides a certificate-based key management infrastructure to support key distribution throughout the internet, to support the first set of mechanisms. This paper describes these mechanisms, as well as the reasons behind their selection and how these mechanisms can be used to provide some measure of security in the exchange of electronic mail.

Index Terms: certificates, cryptography, electronic mail, internet, privacy, security

## 1. Introduction

Probably the best-known use of computer networks is for the transmission of electronic mail. Much less widely known is the inherent lack of security in most mailing systems. Recently this has become the subject of much discussion, and some proposals have been made to enhance the security of electronic mail. The X.411 Recommendation [2], the Message Security Protocol MSP [3], and the protocols in RFC 1113 [4], RFC 1114 [5], and RFC 1115 [6] (which we shall call the *privacy-enhanced mail protocols*), all attempt to address this issue.

This last set of protocols is designed to add security support services to a very large-scale, existing electronic mail structure in a manner transparent to both users and administrators of the network. The most significant contribution of the privacy enhanced mail protocols is to integrate issues of cryptography, systems engineering, key management, and networking on a scale not at-

1. Participation in the meetings leading to these proposals was supported by grant NCC2-397 from the National Aeronautics and Space Administration to the Research Institute for Advanced Computer Science, by grants NAG2-480 and NAG 2-628 from the National Aeronautics and Space Administration to Dartmouth College, and by a Burke Award from Dartmouth College. A preliminary version of this paper was presented at the DIMACS Workshop on Distributed Systems and Cryptography at Princeton, NJ, Oct. 4-6, 1989 [1].

tempted in practice before. The proposals are being implemented by various groups [7][8][9] and are currently draft internet standards.

This paper describes the privacy enhanced mail protocols, which specify a set of protocols for sending electronic mail that provides confidentiality, integrity, and sender authenticity; under certain circumstances, it also provides non-repudiation. The next section presents some background on electronic mailing systems, some relevant aspects of cryptography, and discusses some of the constraints leading to design decisions. Following that, we show how to send a privacy-enhanced message. The fourth section discusses a supporting certificate-based key management architecture, and gives an example of a mechanism used to support that protocol. The fifth and sixth sections consider an alternate key distribution mechanism and the use of mailing lists. Finally, we conclude by comparing these proposals with the X.411 Recommendation and MSP.

This description reflects the state of those protocols as of June 1, 1991, rather than the state specified in the RFCs; however, differences will be noted.[2] Also, we have tied the statements in the text to sections in the RFCs so the reader interested in the reference description (again, except for the differences specified) can quickly and easily locate the relevant text.

We should note in passing that descriptions of earlier versions of these protocols have appeared in the literature [8][10][11]; unlike these, however, the focus of this paper is not only on the protocols themselves but also on the reasons that the choices involved in designing the proposals were made. We hope to provide a broader perspective on some of the design decisions, as well as placing them more completely in context than other work.

## 2. Background and Design Considerations

In this section we review some topics that play a role in enhancing the security of electronic mail. First, we look at a model of message transmittal systems, then we discuss some properties of cryptography, and finally we look at key distribution systems.

### 2.1. Message Handling System Model

Perhaps the most useful model of mail systems is the *Message Handling System Model* [12] (see Figure 1). It treats the mail system as being composed of a number of connected *Message Transfer Agents* and corresponding *User Agents*. A *sender* or an *originator* creates an electronic message and invokes the user agent, which — on behalf of the user — submits that message to a

---

2. Footnotes will describe the specification in the appropriate RFC, and indicate the reason for the change. Revised RFCs have been submitted, and will be released soon; changes that have been made to the existing RFCs are presented in the text and annotated in the footnotes. The reader should be aware that more changes are possible, especially with respect to individual organizations acting as certifying authorities.

message transfer agent. This agent passes the message along to another message transfer agent on another host, which passes it to yet another host, and so forth, until it reaches the message transfer agent at the destination host. This agent passes the message on to a user agent, which saves the message. The *recipient* may then invoke the user agent to read the message, save it, reply to it, or perform other functions.

When a message is accepted by a message transfer agent, it has a particular format the precise description of which depends upon the specific mail system protocol or protocols understood by the agent. When agents attempt to exchange messages, the format of the message must be known to both agents, or it must be translated to a form acceptable to both. For example, message transfer agents handling electronic mail from the largest set of connected networks (called the *internet*) expect messages to be in a format described in RFC-822 [13], which consists of a set of *header fields* followed by a blank line, and then the body of the message. The information in the header lines is used to forward the message from one message transfer agent to another using the Simple Mail Transfer Protocol described in RFC-821 [14]. Another very large network is that of UNIX[3]-based systems, which often use an alternate transfer protocol called UUCP [15] to transmit mail messages by telephone line.

The existence of numerous message transfer agents and protocols suggests that to be used widely, a protocol to enhance security of electronic mail should not require the redesign of existing message transport protocols or message handling systems. The difficulty of coordinating a global change of protocols throughout the internet would be immense, especially since many system managers would delay installing the new software until they were sure it functioned correctly. Thus, compatibility with existing protocols and software would need to be maintained, so there would be two electronic mail protocols, not just one, extant.

Under this assumption, the enhancements must *not* reside at the message transport level,

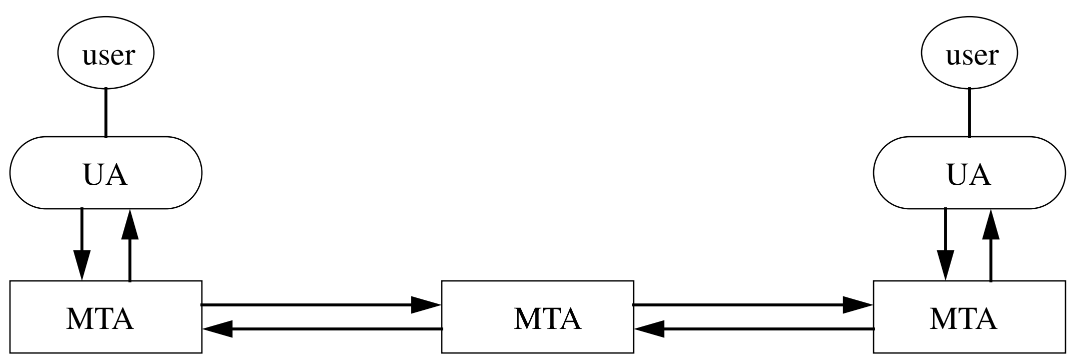


Figure 1. The Message Handling System Model.

3. UNIX is a Registered Trademark of AT&T Bell Laboratories.

but instead must be at the user agent level. The security enhancements should be invisible to the message transfer agent, and therefore the mail systems would be able to send both privacy-enhanced messages and regular messages.

The above suggestion actually has the force of a constraint since the privacy-enhanced mail proposals are intended for widespread use throughout the internet, which is not a centrally-managed network. Instead, it is composed of a collection of autonomous networks, each of which has its own management structure. Any attempt to require sites to alter their message transfer agents would be resisted.

The confidentiality, integrity, and authentication services cannot rely on software over which the user has no control; he or she must have some means of verifying that the software has not compromised the confidentiality or integrity of the message as well as the identity of the sender. Alterations could occur at three points: at the originator's computer, at the recipient's computer, or at any intermediate node. To ensure that the message could not be altered at the first two points would require considering very general issues of operating system security, and certainly require modification of most existing systems. Hence we assume that the sender and recipient can trust the software *above* the message transfer agent level on each other's computer, but that they cannot trust software below that level.[4] This again suggests putting the enhancements *above* the transport level at the end systems as well as using cryptography to protect the messages, since any controls built into the message transfer agents can be subverted (by the system operators, if not by anyone else).

### 2.2. Cryptography for Authentication and Confidentiality

Protecting the confidentiality of messages on a network requires encryption whenever the network is physically beyond the control of the sender. Since he has no power to prevent another from listening in, he must rely on mechanisms that prevent disclosure and enable authentication *even when* a wiretapper is present; ideally, whether that individual is active or passive should be irrelevant.

The literature describing the protection of remote communications suggests using two keys [16]. The first is a one-time key, selected pseudorandomly, called the *session key* or the *data exchange key*; this key is used to encrypt the messages sent during the session. The second key is an *interchange key* associated with the user and/or recipient, and is used to encrypt the data exchange key; the data exchange key is transmitted when the session is begun and then retained for further use. This protocol has three advantages over using the interchange key as a data exchange key.

---

4. Strictly speaking, this is a poor assumption; however, it serves to separate the issues involved in the security of a computer and its attendant software from the security enhancements needed to protect and authenticate an electronic mail message in transit.

If the cryptosystem for the interchange key is symmetric (classical), it is often theoretically possible to derive this key given sufficient ciphertext, especially if the corresponding plaintext is known. For strong cryptosystems, such cryptanalysis requires large amounts of ciphertext (and, possibly, corresponding plaintext); hence, restricting the use of the interchange key to encrypting small amounts of data, and then using different data exchange keys to encrypt (possibly large) amounts of data only once, limits the amount of data a listener will have available to determine the key. Even if the data exchange key is compromised, only the single session will be known; the listener will have to derive a new data exchange key for the next session.

If the interchange key is that of an asymmetric (public key) cryptosystem, then given current technologies, the encryption and corresponding decryption will be more expensive in time, space, or both than those for symmetric cryptosystems. To minimize the overall effect on performance, the short data exchange key can be encrypted using an asymmetric cryptosystem. Hence the trade-off chosen for the privacy-enhanced mail protocols was to use a symmetric data encryption key and an asymmetric interchange key.

Sender authentication and protection of message integrity are provided by sending with a message its *digital signature*. The most straightforward type of digital signature comes from public key cryptography. Encrypting a message with a private key from a public-key cryptosystem that provides authenticity produces a digital signature as large as the message; anyone can validate the signature (as the corresponding public key is widely available), and it is computationally infeasible for an attacker to generate a digital signature for the sender (otherwise, the attacker has the sender's private key).

In practice, it is undesirable to double the size of a message to ensure integrity. For this reason, digital signatures are often produced by using a *manipulation detection code* (also called a *message integrity check*) to compute a small, fixed-size hash of the message, and then encrypting (signing) the hash. These codes, of which one-way (non-invertible) hash functions are examples, must be easy to compute but, as with digital signatures in general, it must be computationally infeasible to find any pair of different files which produce the same manipulation detection code [17].

Both these schemes also provide an integrity check for the message. One can determine the contents of the message, or what the message hashed to, when signed; if the message sent with the signature disagrees or hashes to a different value, then either the message or the signature was altered in transit and the integrity of the communication should be regarded as violated.

Finally, if no one other than the sender has access to the sender's private key, it would not be possible for the sender to disavow a signed message. Should the sender do so, the recipient can prove the message was sent by a party with the sender's private key; since this key should be known only to the sender, the sender would then have to demonstrate that someone had stolen this key.

Note that this assumes a trusted system of some sort, because if that part of the sender's system which handles the private key could be compromised, the private key could be stolen. It also assumes that messages are reliably timestamped, that keys are issued to users whose identities have been authenticated in some fashion (for example, by a public notary), and that all compromised keys are reported at once. (A note of terminology: the key to a classical cryptosystem is called a *secret key* in this paper, whereas the hidden key for a public key cryptosystem is called a *private key.*)

Using cryptography implies that there must be a mechanism for distributing cryptographic keys to communicating parties; we now examine the ways that can be done.

### 2.3. Key Distribution and Management

Creating and distributing interchange keys is a complex problem; one solution described in the literature is to create *key distribution centers*. If the interchange keys are keys for symmetric ciphers, the key distribution centers must be trusted since the cryptographic keys must be resident; hence, there will be at least one individual other than senders and recipients of a message, who could compromise the authenticity and confidentiality of that message (namely, the system security officer or operator of the key distribution center). The practical problems of obtaining the agreement of all autonomous domains in the internet to trust a single key distribution center and its personnel (or a set of key distribution centers) are staggering. Further, symmetric cryptosystems do not provide non-repudiation. Hence the privacy-enhanced mail protocols suggest using an asymmetric cryptosystem for interchange keys, and provide a standard for the management of those keys by encapsulating the required data in *certificates*.

The certificates consist of the user's public key as well as information identifying the associated user and a version number of some form so that, if a user has more than one public key, the sender can identify which one was used. Also a digital signature must be included to protect the integrity of the certificate; this is a hash encrypted using the private key of an issuer, so that anyone may verify the integrity of the data by recomputing the hash and comparing that value with the value obtained by decrypting the stored signature using the stored public key. Note that the key distribution center storing the certificates need not be trusted, since the encrypted hash value binds the user, version, and public key together, and any alterations can be detected by the sender before the key is used; moreover, the private key is not stored at the center. For this reason, that center is referred to as a *directory.*

The privacy-enhanced mail protocols separate the key distribution scheme from the message processing scheme. This way, users operating in an environment where they could trust a central server to manage interchange could do so, whereas users in an environment without such a server could use a certificate-based key distribution mechanism.

## 3. Sending a Privacy-Enhanced Mail Message

The interoperability constraints described in the introduction suggest encapsulating the header fields related to the enhancements and the privacy-enhanced body of the message so that the privacy-enhanced mail message becomes the body of a regular mail message. (We shall refer to the "encapsulated message" to distinguish the privacy-enhanced part of the message from the message in which it is contained.) This has two effects. First, some user agents do not allow users (or other processes) to add special header fields holding the extra information needed to send privacy-enhanced mail;[5] this is not a problem if the relevant header fields are encapsulated in the body. Secondly, as stated in the introduction, an important goal is to be compatible with current electronic mail processing, and adding new header fields requires modifying those programs which must understand all header fields. But if the enhancements are simply part of the body of a message, existing programs need know nothing about the new headers.

The encapsulation mechanism chosen is one widely used in electronic mail throughout the internet [19]; that standard delimits encapsulated material between two delimiter lines. But, as the privacy-enhanced mail protocols allow multiple encapsulated messages to be present either sequentially or with non-privacy-enhanced messages interspersed, using the same delimiter to begin and end encapsulated text introduces ambiguity. So the beginning and ending delimiters must be different. Specifically, each encapsulated message is preceded by a line containing the delimiter

```
-----BEGIN PRIVACY-ENHANCED MESSAGE BOUNDARY----
```

following which come the *encapsulated header fields*, a blank line, and the *encapsulated body*; a line containing the delimiter

```
-----END PRIVACY-ENHANCED MESSAGE BOUNDARY----
```

indicates the end of the encapsulated message.[6] Figure 2 and Figure 3 [7] show examples of encapsulated messages. Note that if the message contains only encapsulated messages, a blank line must separate the first delimiter from the message headers; this is required by [13].

To keep privacy-enhanced mail processing as simple as possible, authentication and integrity checks are applied only to the encapsulated body and neither to the encapsulated header fields

---

5. The *Mail*(1) program in Berkeley UNIX is a good example of this [18].
6. This differs from [4], §4.4, which uses the same line to begin and end an encapsulation.
7. Note that these follow the revisions to the RFCs; in particular, in [4] and [5], all encapsulated header fields began with `X-` to indicate they were experimental and non-standard, as required by [13]. Since they are now part of a draft standard and no longer experimental, the `X-` will be dropped. Also, the version number has changed, and the `Sender-ID` header is now `Originator-ID`.

```
-----BEGIN PRIVACY-ENHANCED MESSAGE BOUNDARY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,3729F9DC6300925A
Originator-ID: someone@somewhere.com::
Recipient-ID: someone@somewhere.com:ptf-kmc:3
Key-Info: DES-ECB,RSA-MD2,
       0F65D99570758593,AEE05B42181E5E261B301291D83DB8F1
Recipient-ID: towho@somewhere.else.com:ptf-kmc:4
Key-Info: DES-ECB,RSA-MD2,
       4B425C85F819327E,0F5FACDB20FD89B9CA8636AE7E70BE74

iwf9eh8CjiF4A9OKf8ZPayOHjAZNLvWYxsGTIr8xQxfQ0uSN4GbnHHzJPXk6saOd
rTLE42WxcsYZsYYZaBfbpBR3GF3nwwIY1O0s3LZTUuwsfsGeIu/v2C+dfSlq/wIE
VesyVXPegFX1MSxZb5bv1J7WK9GE3G58TgwzuYNROPcF61lFq7RDWmS61ji3ZyFy
DNkG/JlAeWlMgP8WpfJgVg==
-----END PRIVACY-ENHANCED MESSAGE BOUNDARY-----
```

Figure 2. Sample encapsulated portion of a message (symmetric interchange keys).

```
-----BEGIN PRIVACY-ENHANCED MESSAGE BOUNDARY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,F8143EDE5960C597
Originator-ID: someone@somewhere.com:MDcMCzACBgNVBAYEAlVUMRIMEAYA
  VQQCEwlUb21Zd2hZcmUMFDAEBgNVBAsEC01YaW4IT2ZZaWNZ:26
Certificate:
 jHUlBLpvXR0UrUzYbkNpk0agV2IzUpk8tEjmF/zxB+bATMtPjCUWbz8Lr9wloXIk
 YbkNpk0agV2IzUpk8tEjmF/zxB+bATMtPjCUWbz8Lr9wloXIkjHUlBLpvXR0UrUz
 agV2IzUpk8tEjmFjHUlBLpvXR0UrUz/zxB+bATMtPjCUWbz8Lr9wloXIkYbkNpk0
MIC-Info: RSA-MD2,RSA,
 5rDqUcMlK1Z6720dcBWGGsDLpTpSCnpotJ6UiRRGcDSvzrsoK+oNvqu6z7Xs5Xfz
Recipient-ID: someone@somewhere.com:MF8xCzAJBgNVBAYTAlVTMRowGAYDV
  QQKEw9Tb21ld2hlcmUsIEluYy4xHjAcBgNVBAsTCkhvbWUgT2ZmaWNl:3
Key-Info: RSA,
 lBLpvXR0UrUzYbkNpk0agV2IzUpk8tEjmF/zxB+bATMtPjCUWbz8Lr9wloXIkjHU
Recipient-ID: towho@somewhere.else.com:MF8xBjAJBgNVBAYTAlVTMRowGA
  YDVQQKEw9FbHNld2hlcmUgQ29ycC4xHjAcBgNVBAsTCFNlY3VyaXR5:4
Key-Info: RSA,
 NcUk2jHEUSoH1nvNSIWL9MLLrHB0eJzyhP+/fSStdW8okeEnv47jxe7SJ/iN72oh

LLrHB0eJzyhP+/fSStdW8okeEnv47jxe7SJ/iN72ohNcUk2jHEUSoH1nvNSIWL9M
8tEjmF/zxB+bATMtPjCUWbz8Lr9wloXIkjHUlBLpvXR0UrUzYbkNpk0agV2IzUpk
J6UiRRGcDSvzrsoK+oNvqu6z7Xs5Xfz5rDqUcMlK1Z6720dcBWGGsDLpTpSCnpot
dXd/H5LMDWnonNvPCwQUHt==
-----END PRIVACY-ENHANCED MESSAGE BOUNDARY-----
```

Figure 3. Sample encapsulated portion (asymmetric interchange keys; [4], p. 19, modified).

nor to the message header fields. Confidentiality of header fields is provided only to the data exchange key and to no other header field. Therefore, header fields containing sensitive information (such as `Subject`) should be put in the encapsulated body, and when necessary can be omitted from the headers; similarly, to check the integrity of header fields, they can be replicated within the encapsulated body. Of course, doing so does *not* ensure the sender named in the replicated header field actually sent the message; the sender can put anything desired into the body of the message. The possibility of confusing the integrity of replicated headers with the authenticity of the sender is sufficient so that the standards do not prescribe the replication of any header fields.[8]

Finally, note that annotated header fields not defined by the privacy-enhanced mail protocols may be interspersed around encapsulated messages to provide related facilities, if appropriate. This enables the user agent to convey information to the user using such fields.

### 3.1. Transforming the Encapsulated Body

In order that encrypted text can be decrypted, and the message integrity check recomputed, correctly despite the vagaries of intermediate mail agents, messages must be transformed into a canonical form before encryption. As this proposal is designed for the internet, the encapsulated message must be sent in a form acceptable to SMTP message transfer agents; the relevant requirements are:[9]

- all characters must be members of the 7-bit ASCII character set; the successful transmission of the eighth (high-order) bit is not guaranteed (and in practice often does not work);
- text lines may be no more than 1000 characters long;
- text lines must be delimited by a carriage return "<CR>" followed by a line feed "<LF>";
- the character sequence "<CR><LF>.<CR><LF>", which is used by the message transfer agents to indicate the end of a message, cannot appear within the body of the message.

Because most computer systems do not use a local representation which complies with this standard (and which in fact varies between computers), some transformation of mail messages is necessary to provide interoperability between hosts using different local representations. Since the encapsulated body of a privacy-enhanced mail message may be encrypted, resulting in characters where the high-order bit must be transmitted accurately for the message to be decrypted correctly, the transformation must take this into account to provide transparency to the underlying message transfer agents.

---

8. [4], §4.4, suggests that under some circumstances replicating header fields for authentication purposes in the body is possible. This has been dropped.
9. [14], §4.5.

Hence the encapsulated body undergoes a three step transformation: first it is put into a machine-independent character-oriented format, then an integrity check is generated and (if required) the message is encrypted, and finally the resulting bit stream is converted to a special set of printable characters suitable for processing by any (reasonable) message transfer agent. When the message text is entered into the system, for example with a text editor, the representation of the characters is that used by the local computer system and is called the *local form*. The local form is then altered in three steps.

### 3.1.1. Canonicalization[10]

The message is converted into a canonical form to ensure that encryption, decryption, and integrity checking all are done on a consistent representation of the message despite the differences between end systems and the vagaries of intermediate mail agents. The representation chosen is to transform characters in the message into their ASCII representations, with the parity (high-order) bit cleared and with line (record) delimiters changed to <CR><LF>; note that this is almost the representation used by SMTP [14].[11] Although any consistent representation would do, this one was chosen because it is a common representation familiar to all SMTP message transfer agents, and therefore software to do this is widely available throughout the internet.

### 3.1.2. Authentication and Encryption[12]

An integrity check is then computed using one of a number of specified algorithms; note that this check can be verified on any type of destination computer because it is computed on the canonical representation of the message and not the local, machine dependent, representation. Currently, three algorithms are specified. The DEA-1-based Message Authentication Code (MAC) algorithm [20] is a well-known message integrity checksum algorithm, has been examined extensively for weaknesses (see for example [21]), and appears to be quite strong; however, it is only suitable when sending a message to *one* recipient and does not provide for non-repudiation [22]. Hence its use is strongly discouraged for messages sent to more than one party. An alternate algorithm, the RSA-MD2 Message Digest Algorithm [6], does not suffer from this weakness and while formal analyses of it are not available, it does not appear to have any exploitable cryptographic weaknesses. Its successor, the RSA-MD4 Message Digest Algorithm [23] also appears to be quite robust and very difficult to compromise so, in the absence of evidence to the contrary, it

---

10. [4], §4.3.2.2.
11. The only difference between this and the SMTP representation is the dot-stuffing transformation ([14], §4.5.2), in which lines consisting only of a period "." have a second period added. This is unnecessary since the purpose of the transformation is simply to ensure a common representation of each character.
12. [4], §4.3.2.3.

| header field | subfields | used … | see section |
|---|---|---|---|
| `Proc-Type` | `protocol,proc_type` | always | 3.1.2. |
| `DEK-Info` | `enc_alg,init_vector` | for encrypted messages | 3.1.2. |
| `Sender-ID` | `address:iss_auth:ver` | always | 3.2.1. |
| `Recipient-ID` | `address:iss_auth:ver` | always | 3.2.1. |
| `Key-Info` | `ik_use,mic_alg,dek,mic` | with symmetric IKs | 3.2.2. |
| `Key-Info` | `ik_use,dek` | with asymmetric IKs | 3.2.3. |
| `MIC-Info` | `ik_use,mic_alg,mic` | with asymmetric IKs | 3.2.3. |
| `Certificate` | `enc_cert` | with certificate-based KM | 4.1. |
| `Issuer-Certificate` | `enc_cert` | with certificate-based KM | 4.1. |
| `CRL` | `crl_encoded` | with certificate-based KM | 4.2.6. |

Figure 4. Summary of header fields; here, "KM" stands for "key management" and "IK" for "interchange key." See the referenced sections for more information on each.

has also been defined as an acceptable algorithm.[13]

If confidentiality is required, the message will then be encrypted using an appropriate encryption algorithm with a data encryption key generated for the message.[14] Currently, the only encryption algorithm allowed is the DES in cipher block chaining mode [24][25]. This U. S. government standard is in widespread use, has withstood the test of time, can be implemented very efficiently in either hardware or software, and all attacks known are very time-consuming.[15] It transforms plaintext blocks of 8 bytes (64 bits) to ciphertext blocks of 8 bytes, and so requires that the message length be a multiple of 8 bytes. So the mail message is padded by creating a byte containing the number of extra octets needed, and replicating it appropriately. For example, if the message required 3 bytes of padding, the bits "030303" (in hex) would be added. If the mail message length is a multiple of 8 bytes, 8 bytes of padding, each containing "08" (hex), are added for consistency.[16]

Figure 4 summarizes the encapsulated header fields that convey the information needed to decrypt, and check the integrity of, the message. All encapsulated messages contain the encapsulated header field

13. Other integrity check algorithms may be added later; see [6], §1 and §4. For example, RSA-MD4 was added after [6] was issued.
14. [4], §4.6.1.1. This section also provided for partial encryption, in which only portions of the message were encrypted. Partial encryption has since been dropped from the protocol.
15. Again, other message encryption algorithms may be added later; see [6], §1 and §2.1.3.
16. This differs from [4], §4.3.2.3, which required padding with bytes of all bits set; the high-order bit being set disambiguated the padding from the message. However, as the message integrity checksum (which contains arbitrary binary data) is also encrypted using the DES in CBC mode, another form of padding must be defined for it. The new padding can be used for both the message and the message integrity checksum.

| protocol described in | `protocol` subfield value |
|---|---|
| RFC-989 | 1 |
| RFC-1040 | 2 |
| RFC-1113 | 3 |
| *next RFC* | 4 |

Figure 5. Table of Protocol Numbers.

---

`Proc-Type:` *`protocol_version,proc_type`*

which indicates the type of processing done. Its first subfield, *protocol*, is the number of the protocol used (see Figure 5). Its second subfield, *proc_type*, is `MIC-ONLY` if *no* part of the encapsulated message is encrypted, `ENCRYPTED` if the encapsulated message is encrypted, or `MIC-CLEAR` if the encapsulated message is not encrypted and not encoded as described in section 3.1.3.[17] A second header field, required only for encrypted encapsulated messages, conveys the message encryption algorithm and initialization vector:

`DEK-Info:` *`encrypt_alg,init_vector`*

The first subfield is a string indicating how the encapsulated message was encrypted; as only the DES in cipher block chaining mode is defined for use, this string must be `DES-CBC`. The second subfield is the initialization vector. Although the initialization vector is usually encrypted to prevent spoofing by altering key bits (and also altering the decryption of the first block), the integrity check will detect such tampering; hence the initialization vector need not be encrypted and is represented as a string of 16 hexadecimal digits. The integrity check and data encryption key will be transmitted after being encrypted with the interchange key, and we shall defer the details until the next section.[18]

### 3.1.3. Printable Encoding[19]

After the second step, the message text should be thought of as a bit stream. (The encapsulated headers are not part of this stream unless they are replicated within the message text for integrity checking purposes.) Note that even though there are an integral number of characters, all 8 bits may be significant. This form must be changed into another that meets the SMTP requirements stated above. The simplest way to do this is to expand the message.

The first thought is to group the bits into sets of 7, and transmit the 7-bit ASCII character corresponding to each. However, since some of those character sequences are special to SMTP (such as "<CR><LF>.<CR><LF>") and others to the privacy enhancements (such as "--", which oc-

17. [4], 4.6.1.1. That section provides only for the field values `MIC-ONLY` and `ENCRYPTED`.
18. [4], §4.6.1.2.
19. [4], §4.3.2.4.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | A | 8 | I | 16 | Q | 24 | Y | 32 | g | 40 | o | 48 | w | 56 | 4 |
| 1 | B | 9 | J | 17 | R | 25 | Z | 33 | h | 41 | p | 49 | x | 57 | 5 |
| 2 | C | 10 | K | 18 | S | 26 | a | 34 | i | 42 | q | 50 | y | 58 | 6 |
| 3 | D | 11 | L | 19 | T | 27 | b | 35 | j | 43 | r | 51 | z | 59 | 7 |
| 4 | E | 12 | M | 20 | U | 28 | c | 36 | k | 44 | s | 52 | 0 | 60 | 8 |
| 5 | F | 13 | N | 21 | V | 29 | d | 37 | l | 45 | t | 53 | 1 | 61 | 9 |
| 6 | G | 14 | O | 22 | W | 30 | e | 38 | m | 46 | u | 54 | 2 | 62 | + |
| 7 | H | 15 | P | 23 | X | 31 | f | 39 | n | 47 | v | 55 | 3 | 63 | / |

Figure 6. Printable Encoding Characters

curs in the encapsulation delimiters), there would need to be an escape sequence. It is easier to ignore this issue by restricting the set of characters to those which are not special. The letters and digits fall into this category, as do numerous punctuation characters, but there are fewer than 128 ($2^7$) of them. Hence each bit stream is grouped into sets of 6 bits, each of which is mapped into a character in the alphabet shown in figure 6.[20]

The final set of bits in each stream may contain 2 or 4 bits rather than 6. In the former case, the two meaningful bits are padded with four cleared bits, and two "=" characters are appended to indicate the last 4 bits are padding. In the latter case, the pad is two cleared bits, and one "=" is appended. Finally, the resulting character stream is split into lines of 64 printable characters (except, possibly, for the last line, which may contain fewer).[21] This form is suitable for transmission by any SMTP-like message transfer agent.

Grouping the stream into sets of 4 bits rather than 6 would make encoding faster and eliminate the special handling of the final set of bits. However, this would also double the length of the encoded message. With most messages, network transmission times will dominate the time to encode the message, so the 6-bit encoding was chosen.

Because reading unencrypted, integrity-checked encapsulated messages encoded in this format would require all user agents to be able to translate this printable encoding, the privacy-enhanced mail protocols allow this step to be skipped for such messages; the second field in the `Proc-Type` encapsulated header field will be `MIC-CLEAR` for such messages. However, as `MIC-CLEAR` messages are not printably encoded, privacy-enhanced user agents should indicate that failure of the integrity checking for this type of message may result from differences in character representations between the end hosts and transformations by the message transfer agents

20. This is actually a subset of the International Alphabet IA5; the elements of this subset are represented identically in IA5 and ASCII. See [4], p. 13.

21. [4], §4.3.2.4 also provided that, if the message were partially encrypted, an asterisk "*" was to be placed before and after the character sequences corresponding to cleartext regions. As partial encryption has been dropped, so has this character.

rather than an alteration of the contents of the encapsulated message.[22]

### 3.2. Transmission of Integrity Check and Data Encryption Key

We now turn to the transmission of the data encryption key and the message integrity check. The former must be encrypted using an interchange key to protect the confidentiality of the encapsulated message body, the latter to ensure authenticity and integrity. The mechanism used to do this depends on whether the interchange key is used with a classical cryptosystem or with a public key cryptosystem. In either case, the interchange key depends upon proper identification of the sender and recipient.

#### 3.2.1. Sender, Recipient, and Interchange Key Information[23]

The sender and recipient of each message are identified in the following encapsulated header fields:[24]

```
Originator-ID: entity_id:issuing_authority:version
Recipient-ID: entity_id:issuing_authority:version
```

These fields specify which interchange key was used. The first subfield contains the identity of the sender or receiver, the second the identity of the authority issuing the interchange key, and the third an indicator of the specific interchange key being used. Since interchange keys depend on both the sender and the recipient, each `Recipient-ID` field is associated with the last preceding `Originator-ID` line.

The first subfield is mandatory; it also requires that each sender and recipient be uniquely identifiable. Hence it assumes the form *user@host*, where *user* is unique to *host* and *host* is unique throughout the set of hosts using electronic mail. Any scheme guaranteeing this will work, but those sites which transmit into the internet should use the fully qualified domain name for *host*;[25] that name will be processed in a case-insensitive manner.[26]

The second subfield contains the unique name of the authority that issued the interchange key; it need not be unique among all entities, but must be unique over all issuing authorities.[27] If certificate-based asymmetric key management is used, then this name is to be the issuing authority's Distinguished Name encoded using the basic encoding rules of ASN.1 [26][33] and represent-

22. This processing mode is new and not described in [4], 4.6.1.1.

23. [4], §4.6.2.1, §4.6.4.1.

24. In [5], the `Originator-ID` header field was called the `Sender-ID` header field, but the name was changed to distinguish it from the more common `Sender` field ([13], §4.4.2).

25. [4], §5.2.1.1, §6.1.

26. This has been added to conform to the normal way of processing host names in the internet.

27. [4], §5.2.1.3.

ed using the printable encoding form described in section 3.1.3.[28] This convention was chosen because it is also used in certificates to identify issuing authorities unambiguously. The nature of a Distinguished Name will be described later; however, as examples, in Figure 3 the issuing authorities of the certificates of the recipients are the Home Office of Somewhere, Inc. and Security of Elsewhere Corp., both in the United States.

The third subfield is some number or string selected by the issuing authority to disambiguate among the interchange keys issued by that authority to the sender/recipient pair. For certificate-based interchange key management schemes, this subfield must be the serial number of the certificate; for other schemes, the proposal recommends the use of timestamps, which not only provide uniqueness but also allow an expiration date to be prescribed (for example, two years after issue).[29]

If the subfields contain redundant information, they may be omitted.[30] For example, if the interchange key is used in a symmetric cryptosystem, the contents of the last two subfields of both `Originator-ID` and `Recipient-ID` encapsulated header fields will be the same for each sender/recipient pair, and so are normally omitted from the `Originator-ID` field. In Figure 2, the encapsulated header field

```
Originator-ID: someone@somewhere.com::
```

identifies the sender as user *someone* at host *somewhere.com*, but leaves the issuing authority and version subfields blank. Additional relevant information is in the associated `Recipient-ID` line:

```
Recipient-ID: someone@somewhere.com:ptf-kmc:3
```

means that the recipient is *someone@somewhere.com*, that the interchange authority named *ptf-kmc* has issued an interchange key for *someone* to use when he sends messages to himself (since the last preceding `Originator-ID` field identified *someone@somewhere.com* as the sender), and that this key can be identified uniquely by the string `3`. The second such field,

```
Recipient-ID: towho@somewhere.else.com:ptf-kmc:4
```

28. The common name attribute (CN) should be omitted, as the issuing authority is always an organization or an organizational unit. Originally, any unique name could be used, but in an internet without a central administrator, this could lead to ambiguities. Also, in [4], the Distinguished Name had to be written using the method in [27] before being put into ASN.1; this requirement has been dropped as unnecessary.

29. [4], §5.2.1.3. Originally any disambiguating string could be used for any scheme, but since certificates are issued with a serial number unique to each issuing authority, it is simplest to make the identifying string identify the precise certificate. Note that certificates contain expiration times, so those are as readily available as if they were given as the identifying string.

30. [4], §4.6.2.1.

means that *ptf-kmc* has also issued an interchange key for *someone@somewhere.com* to use when sending messages to *towho@somewhere.else.com*, and that this key can be identified uniquely by the string `4`.

Similarly, if the sender's public key is sent in the message as a certificate containing both the version number and issuing authority (using the certificates and the `Certificate` header field described in the next section), the last two subfields in the `Originator-ID` encapsulated header field may be omitted. In Figure 3, the certificate in the `Certificate` line is that of the sender in the preceding `Originator-ID` line, so the last two subfields of that field are omitted. But the `Recipient-ID` lines specify the issuer and serial number of the recipients' certificates, so that they can determine which of their public keys was used to encrypt the data encryption key.

If a message is sent to multiple recipients, it must of course have multiple `Recipient-ID` fields. But notice that under certain circumstances a single message may have more than one `Originator-ID` field. For example, if a sender shares a symmetric interchange key with one recipient, and another recipient uses an asymmetric interchange key, then to send both the same letter, the sender must identify two different interchange keys using distinct `Originator-ID` fields.

### 3.2.2. Symmetric Interchange Keys

With symmetric cryptosystems, the encryption key is either the same as, or easily derivable from, the decryption key. For this reason, the *interchange key* is defined as the single key associated with both the sender and the recipient, and the message integrity check and data encryption key follow the identification of each sender and recipient.[31]

The message integrity check and data encryption key for each sender and recipient pair are given in encapsulated header fields of the form[32]

`Key-Info:` *`ik_use`*`,`*`mic_algorithm`*`,`*`dek`*`,`*`mic`*

One such line normally follows each recipient name, and uses the interchange key associated with the last-preceding sender and that recipient. The first subfield identifies the algorithm which was used to encrypt the data exchange key; the proposal requires using either the DES in electronic code book mode (indicated by `DES-ECB`) [25] or the DES in encrypt-decrypt-encrypt mode (indicated by `DES-EDE`) [28]; these were chosen because they are cryptographically very strong, can be implemented efficiently in hardware or software, and are used by standards relied upon throughout different communities.[33] The second subfield identifies the algorithm used to generate the mes-

31. [4], §5.2.

32. [4], §4.6.4.2.1.

33. [6], §1, §2.1.1, and §2.1.2. Other data exchange key encryption algorithms may be added later.

sage integrity check; this subfield must be `MAC` (for the DEA-1-based algorithm), `RSA-MD2` (for the RSA-MD2 Message Digest Algorithm) or `RSA-MD4` (for the RSA-MD4 Message Digest Algorithm).[34] The third and fourth subfields are the data exchange key and the message integrity check encrypted using the algorithm given by the first subfield with the interchange key. Both are represented as strings of hexadecimal digits. For example, in Figure 2, the encapsulated header field

```
Key-Info: DES-ECB,RSA-MD2,
  0F65D99570758593,AEE05B42181E5E261B301291D83DB8F1
```

indicates that the message integrity check was computed using the RSA-MD2 Message Digest Algorithm, and both it and the data exchange key were encrypted using the interchange key and the DES algorithm in electronic code book form. The second `Key-Info` encapsulated header field in that figure indicates the same algorithms, but since another `Recipient-ID` line has occurred since the last `Key-Info` line, the data encryption key and message integrity check were encrypted with the interchange key for the sender/recipient pair *someone@somewhere.com* and *towho@somewhere.else.com* (and hence look quite different than in the preceding line).

### 3.2.3. Asymmetric Interchange Keys

With public key cryptosystems, encryption and decryption keys are different, and neither can be derived merely by knowing the other. This allows authentication and encryption to be done with two different keys, and so the *interchange key* is comprised of both the recipient's public key and the sender's private key, the former being used to encrypt the data exchange key (confidentiality), and the latter being used to encrypt the message integrity check (integrity/authenticity).[35] The message integrity check and data encryption keys are given by two different encapsulated header fields because the parts of the interchange key differ. The message integrity check is given in encapsulated header fields of the form[36]

`Key-Info:` *`ik_use`*`,`*`dek`*

The first subfield is the name of the algorithm used to encrypt the data encryption key; since only one algorithm is currently defined for this purpose, the first subfield will always be `RSA`.[37] The second subfield is the data exchange key, encrypted using the recipient's public key and represented as a string of printable encoding characters using the encoding transformation. For example, in Figure 3 the encapsulated header fields

---

34. [6], §1, §4. Again, other message integrity check algorithms may be added later.

35. [4], §5.2.

36. [4], §4.6.4.2.2.

37. Other data exchange key encryption algorithms may be added later; see [6], §1 and 3.1.

```
Recipient-ID: someone@somewhere.com:MF8xCzAJBgNVBAYTAlVTMRowGAYDV
   QQKEw9Tb21ld2hlcmUsIEluYy4xHjAcBgNVBAsTCkhvbWUgT2ZmaWNl:3
Key-Info: RSA,
 lBLpvXR0UrUzYbkNpk0agV2IzUpk8tEjmF/zxB+bATMtPjCUWbz8Lr9wloXIkjHU
Recipient-ID: towho@somewhere.else.com:MF8xBjAJBgNVBAYTAlVTMRowGA
   YDVQQKEw9FbHNld2hlcmUgQ29ycC4xHjAcBgNVBAsTCFNlY3VyaXR5:4
Key-Info: RSA,
 NcUk2jHEUSoH1nvNSIWL9MLLrHB0eJzyhP+/fSStdW8okeEnv47jxe7SJ/iN72oh
```

show that the sender (*someone@somewhere.com*) has encrypted the data encryption key using the RSA cryptosystem and his public key for the first recipient (himself), and using RSA and *towho@somewhere.else.com*'s public key for the second.

The message integrity check is transmitted similarly, in an encapsulated header field of the form[38]

MIC-Info: *ik_use*,*mic_algorithm*,*mic*

The first subfield identifies the algorithm used to compute the integrity check, and so must be either `MAC` (if the DEA-1-based algorithm was used), `RSA-MD2` (if the RSA Message Digest Algorithm was used), or `RSA-MD4` (if the newer RSA Message Digest Algorithm was used).[39] The second subfield identifies the algorithm used to encrypt the integrity check; again, currently this must be `RSA`. The final subfield is the message integrity check encrypted using the sender's private key and then represented as a string of printable encoding characters using the encoding transformation. If the message is encrypted, then the message integrity check must also be encrypted (using the same algorithm and data encryption key as the message); otherwise, if an encrypted message known to correspond to one of *n* possible plaintexts, were intercepted, then comparing the intercepted message's integrity check with those of the plaintexts would show precisely which one the intercepted message corresponded to.[40] Note that the information is associated with the sender and not the recipient, so this line should only occur after the `Originator-ID` encapsulated header field. In Figure 3, the encapsulated header field

```
MIC-Info: RSA-MD2,RSA,
 5rDqUcMlK1Z6720dcBWGGsDLpTpSCnpotJ6UiRRGcDSvzrsoK+oNvqu6z7Xs5Xfz
```

and its location show that the message integrity check was computed using the RSA-MD2 Message Digest Algorithm, and encrypted using the RSA algorithm and *someone@somewhere.com*'s private key.

38. [4], §4.6.3.1.

39. Other message integrity check algorithms may be added later; see [6], §1, §4.1, and 4.2.

40. This changes [5], §4.6.2.3, in which the signed message integrity check is never encrypted.

## 4. Key Management Infrastructure

If asymmetric interchange keys are used, a method for ensuring that correct public keys are made available must be provided; without such assurance, an active wiretapper could spoof the intended recipient and read the sender's supposedly private message, or could alter or forge messages from another sender. The privacy-enhanced mail protocol suggests using X.509 [29] compliant certificates containing an entity name, a public key, an issuer name, and a digital signature created by the issuer based on its public key and the information in the certificate. This section presents the architecture for managing those certificates.

### 4.1. Overview of Certificate Management Architecture

The architecture envisions a set of trees with the inner nodes being organizations entitled to issue certificates. The root of each tree will be a *top-level certification authority*; each of the inner nodes will be a *certification authority*. The leaves are subjects to whom certificates are issued, and who are not authorized to issue certificates themselves. The next higher tier is composed of organizations who may certify users; the next tier is composed of organizations who may certify other organizations, and so forth. In all cases a certifying authority may only certify those entities directly beneath it. The top-level certifying authority (or any certifying authority, for that matter) may establish its own procedures for and rules for determining what entities it will certify, how those entities request certification, and how the certification is done.[41] This arrangement of certifying authorities within each tree is a subset of the certification hierarchy allowed under X.509.

The problem of transmitting and obtaining certificates for use with privacy-enhanced electronic mail in the absence of ubiquitous directories containing certificates is solved by including one or more instances of two optional encapsulated header fields in the privacy-enhanced mail message. The encapsulated header field[42]

`Certificate:` *`encoded_certificate`*

contains the sender's certificate as a bit stream represented as a string of printable encoding characters using the encoding transformation. Normally only one such field would be present. Similarly, the encapsulated header field[43]

`Issuer-Certificate:` *`encoded_certificate`*

is like the `Certificate` field, except that it contains the certificate of the issuer of a certificate. As many of these encapsulated header fields as necessary to enable the recipient to validate the certificates in either the `Certificate` or in another `Issuer-Certificate` header field may be

41. [5], §3.1.
42. [4], §4.6.2.1.
43. [4], §4.6.3.1.

present. Note that this field is optional, and once directory servers containing certificates become widely available, will fade into disuse. Examples of both these fields are given in Figure 3.

We now examine the contents of certificates, the management of certificates, and how the two relate to one another.

### 4.2. Certificate Definition and Use

A *certificate* binds a public key to an issuing authority, subject (which may be a user or some other entity), and other information. Each contains a version number indicating which certificate format is used (currently the only format defined is this one, so the version number will be `0`),[44] a subject identity, an issuer name, a serial number defined by the issuer (certifying authority) and unique to that certifying authority,[45] a validity period during which the certificate is valid,[46] the subject's public key, and a certificate signature binding all the above information together.[47] The signature is generated by computing a hash of the certificate and encrypting the hash with the certifying authority's private key. Currently the only signature algorithm is the `md2WithRSA` algorithm [30], which computes a message hash using the RSA Message Digest Algorithms RSA-MD2, and then encrypts it using the RSA encryption algorithm with the private key of the issuing authority.[48] An example certificate is shown in figure 7.

#### 4.2.1. Identity and Distinguished Names

Both issuers and subjects must be identified unambiguously; the mechanism chosen to do this is the *Distinguished Name*,[49] which is a set of attributes containing a key and a value. An example of a typical Distinguished Name might be:

/C=US/O=Dartmouth College/OU=Dept. of Math & CS/CN=Matt Bishop/

---

44. [5], §3.4.1.1.
45. [5], §3.4.1.2.
46. [5], §3.4.1.5.
47. [5], §3.4.1.
48. Revisions to separate the signature algorithms from the hash algorithms; this particular signature algorithm is the same as given in [4], §3.1.4.7; [6], §4.2, but in those RFCs it is defined as a hash algorithm followed by an encryption algorithm. As other equally strong (or stronger) signature functions are found that are computationally as efficient as this one, they may be added to the list of acceptable hash algorithms. A signature algorithm using RSA-MD4 was not added as it is quite new, and so the protocol authors were less comfortable using it to protect the interchange keys embodied in the certificates.
49. The Distinguished Name may be written using the full T.61 character set, as described in X.500 [31]; however, some alternate representations of characters which software cannot print would be represented as a backslash followed by the character's octal representation.

```
30 82 01 94
   30 82 01 32                  { version, serial numbers (defaults 0)}
      02 01 01                  { serial number (1) }
      30 0a                     { signature information }
         06 04 55 08 03 02            { algorithm (MD2 hash, RSA signature) }
         02 02 02 78                  { parameter (632) }
      30 49                     { issuer's distinguished name }
         31 0b
            30 09
               06 03 55 04 06      { country }
               13 02 55 53         { US }
         31 1a
            30 18
               06 03 55 04 0a      { organization }
               13 11 44 61 72 74 6d 6f 75 74 68 20 43 6f 6c 6c 65 67 65    { Dartmouth College }
         31 1e
            30 1c
               06 03 55 04 0b      { organizational unit }
               13 15 44 65 70 74 2e 20 6f 66 20 4d 61 74 68 2e 20 26 20 43 2e 53 2e
                                                                            { Dept. of Math. & C.S. }
      30 1a                     { validity period }
         17 0b 39 30 30 33 31 36 31 32 30 30 5a    { from 3/16/90 at 12:00 GMT }
         17 0b 39 32 30 33 31 36 31 32 30 30 5a    { to 3/16/92 at 12:00 GMT }
      30 5f                     { subject's distinguished name }
         31 0b
            30 09
               06 03 55 04 06      { country }
               13 02 55 53         { US }
         31 1a
            30 18
               06 03 55 04 0a      { organization }
               13 11 44 61 72 74 6d 6f 75 74 68 20 43 6f 6c 6c 65 67 65    { Dartmouth College }
         31 1e
            30 1c
               06 03 55 04 0b      { organizational unit }
               13 15 44 65 70 74 2e 20 6f 66 20 4d 61 74 68 2e 20 26 20 43 2e 53 2e
                                                                            { Dept. of Math. & C.S. }
         31 14
            30 12
               06 03 55 04 03      { common name }
               13 0b 4d 61 74 74 20 42 69 73 68 6f 70 69 73 68 6f 70      { Matt Bishop }
      30 59                     { subject's public key information }
         30 0a
            06 04 55 08 03 02      { algorithm (RSA signature, MD2 hash) }
            02 02 02 00            { length (512; here, the modulus size in bits) }
         03 4b 00 30 48            { modulus }
                  02 41 00 cf 77 fd 00 aa e3 46 2c 66 ee 7c 2b fd de 2d 09 ed 2a 3e dd 3c a1 93 98 68
                        f9 95 96 8e 17 9e a8 92 8c 44 7c b8 4f 92 fe 02 7c ab d5 09 9a ff 8d b8 1e f2
                        f4 80 b0 6d e8 30 fa 62 ca 09 90 f7 e5
                                   { modulus(10866017727492218440042295274990800056806083-
                                            243773962467509491311256998916397863613143571-
                                            763061020265939911467248114019931398403135074-
                                            478528619036491708389) }
                  02  03  01  00  01 { exponent (65537) }
   30 0a                        { signature }
      06 04 55 08 03 02            { algorithm (MD2 hash, RSA signature) }
      02 02 02 78                  { parameter (632) }
   03 50 00 42 e3 50 db c2 1b da a1 cd 0d 2b 70 e6 66 5f 59 29 91 dc 28 76 df 2a ef a1 7a c2 7d 13 99
         e0 ea e8 d4 7e 5a 18 70 0e 73 b8 c1 cc 68 98 27 a6 f2 aa f2 d5 0b 43 69 a0 64 16 92 8a 7c ed
         cd e3 45 b0 8e 54 a8 06 36 24 3e 04 f0 28 84 20 75 a1       {digital signature }
```

Figure 7. An example of a certificate. The certificate is a bit stream encoded as described in section 3.1.3. For illustrative purposes, the hex representation of the ASN.1 encoding is shown above. The corresponding issuer public key is 3186 3884 a5a5 6a53 7a8c 5c79 801c 63fd 9851 1336 ae14 de61 f8e2 1bf0 81a2 1cbd 0268 6f63 ea2a 9f8c c18d 1a0d c6f7 a10f 748d 9d90 5b0a 5056 d057 93bb 3776 9aa9 2bef a916 a8dc 2099 fbda 1acb e991 ae00 (hex).

| *attribute* | *meaning* | *1* | *2* | *3* | *4* | *5* |
|---|---|---|---|---|---|---|
| C | country (ISO 3166 [34] encoding) | m* | m | o | o | o |
| S | state or province name | o | m | o | o | o |
| L | locality name (*e.g.*, city) | o | m | o | o | o |
| O | organization name | m* | - | m | m | o |
| OU | organizational unit name | o | - | o | o | o |
| CN | common name | - | m | m | - | m |
| T | title | - | - | o | m | - |
| PA | postal address (*e.g.*, street or box) | o | m | o | o | o |

Figure 8. Summary of Distinguished Name attributes and their meanings. The columns labelled 1 through 5 indicate whether, for that kind of Distinguished Name, the attribute is mandatory (m), optional (o), or prohibited (-). The columns refer to the following kinds of Distinguished Names: (1) certifying authority; (2) residential person; (3) organizational person; (4) organizational role; (5) distribution list.

---

where "Matt Bishop" is the subject's Common Name,[50] "Dept. of Math & CS" is the Organizational Unit name, "Dartmouth College" the Organizational name, and "US" the Country name.[51] The allowed Distinguished Name attributes and what they signify are shown in Figure 8; precisely which attributes may be used depends on which kind of Distinguished Name is being used:[52]

A *certifying authority* is an entity authorized to issue certificates, and the Distinguished Name of such an authority *always* appears in the issuer field. As such an authority is *never* an individual, it has no associated Common Name or Title.[53] Further, the Country in which the certifying Authority resides (or the Organization Name, if it is a multinational entity), must be present. Further identifying information (such as the Locality, Organizational Unit, and so forth) must be present if necessary to unambiguously identify the issuing authority.

A *residential person* is an entity not claiming affiliation with any organization, for example a person who obtains a certificate for private use. The Distinguished Name must unambiguously

50. Originally this was to be a structured Personal Name component (as defined in X.400 [12]) but was changed to conform to X.500.

51. For compatibility with the U. S. Government Open Systems Interconnection Profile [32], [5], §3.4.1.3 limited the number and length of each field in the Distinguished Name. This has since been changed to conform to the limits in X.520.

52. Note that [5] allows any Distinguished Name attributes to be used, unless expressly prohibited (and this is done only in the case of certifying authorities, which cannot have the Common Name attribute). However, allowing a certificate with an organization in the subject's Distinguished Name to be issued to an affiliated user would be very misleading unless the certificate were examined carefully, so the proposal was revised to eliminate possible confusion.

53. [5], §3.4.1.4.

identify the person by providing an address. No organizational attributes may be given.

An *organizational person* is an entity claiming affiliation with an organization, for example a professor whose certificate is issued by (or on behalf of) his college. In its Distinguished Name, the Common Name of the person and the Organization with which he is affiliated are required; other attributes are optional.

An *organizational role* is a position within an organization to which the certificate is tied; it may be filled by many people over time, but the certificate stays bound to the role and not the individual holding that position. Required attributes of this kind of Distinguished Name are the Organization and the Title of the role; other attributes except the Common Name are allowed. The Common Name is specifically disallowed to prevent confusion between this type of Distinguished Name and that of an organizational person.

A *distribution list* is a collection of users, and mail sent to it is forwarded to all those users. Its Distinguished Name requires a Common Name attribute, and it is recommended that a second, terminal Common Name attribute with value the distinguished string `Distribution List` also be present (to distinguish the Distinguished Name from that of an individual). Any other attributes except Title may be present.

As another example, the Distinguished Name for the organizational role of postmaster might be:

/C=US/O=USRA/OU=RIACS/OU= Network Systems Division/T=Postmaster/

This name contains two hierarchical organizational unit names ("Network Systems Division" is a branch of "RIACS" which is in turn a research laboratory of "USRA"). If multiple organizational unit names are present, they are taken hierarchically, with the broadest unit first.[54]

This information is encoded into a certificate using the ASN.1 representation shown in X.509, Annex G, which is part of the international directory standard. This is the same format as is used for the encoding of the issuer identity in the `Originator-ID` and `Recipient-ID` lines described in section 3.2.1.

### 4.2.2. Certification Authorities

At the root of each certification hierarchy is a top-level certifying authority. This organization may issue certificates and it may authorize other organizations to issue certificates, possibly imposing stringent technical and managerial requirements to ensure the security and integrity of the certificate signing procedure. It may also make agreements with other top-level certification authorities to allow interoperation across different hierarchies; these agreements are essentially cross-certification, where each top-level certification authority generates and makes available certificates

54. [5] does not specify the order of organizational units, but they are to be written most significant member first.

for the other.[55]

Cross-certification agreements are not transitive; that is, if one top-level certifying authority cross-certifies a second which in turn cross-certifies a third, the first and third are *not* cross-certified. The reason for this restriction is that cross-certification implies satisfaction with and acceptance of legal and certification procedures of the cross-certified top-level certification authority, and that must be done with knowledge of those agreements, not by default. Hence, a certification path may have elements of at most two hierarchies.

A certifying authority is one which may issue certificates. With three exceptions to be discussed in the next section, when a certificate is issued, the issuing certifying authority is vouching for the identity as embodied in the subject's Distinguished Name; that is, if an organizational affiliation is shown, such affiliation has been proved, and the public component as embodied within the certificate belongs to that subject. How such proof is given is up to the certifying authority, but may be affected by contracts or licenses with the top-level certifying authority.[56]

A key component of acquiring this proof is the *organizational notary.*[57] The organizational notary is a clearinghouse for requests for certificates within an administrative domain such as an organization or organizational unit, and is assumed to be somewhat independent of the users in that administrative domain. Only those users in the domain may obtain certificates through that organizational notary, who accepts the requests for certificates and validates the information to be put in the certificate; the manner in which this is done is up to the organization, except for certain minimum requirements which the top-level certifying authority may establish as a condition of making the organization a certifying authority. If necessary, the notary may change the period of validity requested (to comply with the policies of the certifying authority), the serial number requested (to ensure it is unique among all those issued by the certifying authority), and any other field in the certificate except the user's personal name (in the Distinguished Name) and the user's public key. Under normal circumstances, though, only the serial number and validity period are likely to be changed.

The use of an organizational notary has benefits both to the user and to the organization. First, it enables the organization to validate those to whom it issues certificates quickly and easily. Secondly, the user need not go to an outside party to prove his or her association with the organization; it can be handled in-house. Third, should the user leave the organization, his or her affiliation can be repudiated by revoking the certificate. The benefits to the certifying authority are also large, residing mainly in the ability to delegate the job of verifying the information provided by the

55. [5], §3.3.3.1.

56. [5], §3.3.1.

57. [5], §3.3.2.

user. Since there will be far fewer organizational notaries than users, they can be screened more carefully.[58]

A certifying authority's private key is used to sign certificates and is *never* used as a component of an interchange key, hence a certifying authority's certificate is useful only for validation purposes. This limits the damage should such a private key be compromised; while certificates could be forged (until the corresponding public key certificate were revoked), privacy-enhanced messages protected using certificates issued before the compromise by the compromised certifying authority would still be protected.[59] (Of course, an attacker could use the private key to generate forged certificates and date them before the compromise; hence, all certificates issued by the compromised certifying authority should be considered suspect when its private key is compromised.)

Finally, even though there may be other certifying authorities between the issuer and the top-level certifying authority, all certificate hierarchies are expected to follow the convention that each certifying authority's certificate is signed by the top-level certifying authority. This shortens the certificate validation procedure, described below.[60] This will be particularly useful during the initial deployment, when directories of certificates are not ubiquitous, as at most three certificates per sender will have to be included in a privacy-enhanced message.

### 4.2.3. Conventions and Special Certificates[61]

Under normal circumstances, a person obtaining a certificate with an associated organization in the subject name is presumed to be closely affiliated in some way with that organization. Under certain circumstances, such affiliation may not be desired, for example if the person is visiting for a limited period of time. In this case, the certifying authority may issue a *guest certificate* in which the final attribute of the issuing authority name is an Organizational Unit name with a value the distinguished string `GUEST`. For example, if the certifying authority Dartmouth College issues a certificate to visiting professor Tom Jones, the issuing authority name would be

/C=US/O=Dartmouth College/OU=GUEST/

and the subject name would be

/C=US/O=Dartmouth College/OU=GUEST/OU=Dept. of Math & CS/CN=Tom Jones/

---

58. [5], §3.3.3.
59. [5], §3.1.
60. [5] does not require top-level certifying authorities to sign certificates of all certifying authorities in their domain.
61. [5] combines the `GUEST` and `NOTARY` certificates, calling them notary certificates. The distinction was made to clarify whether or not organizational affiliation as well as identity was being vouched for.

Note that a guest certificate affirms the identity of the user; it merely asserts a weak, rather than a strong, affiliation with the organization. Of course, the certifying authority cannot issue such a certificate for a member of another organization.

A similar type of certificate, the *notary certificate*, asserts that the user's identity has been proved to the certifying authority but that no organizational affiliations are present. This is most useful when certificates are issued to residential persons, and in fact notary certificates may only be used for that type of entity; they my never be used for an organizational person or role (nor certifying authority or distribution list). In this case, the final attribute of the issuing authority name is an Organizational Unit name with a value the distinguished string `NOTARY`. For example, if Dartmouth College issues a residential certificate to Tom Jones, the issuing authority name would be

/C=US/O=Dartmouth College/OU=NOTARY/

and the subject name might be

/C=US/S=New Hampshire/L=Hanover/CN=Tom Jones/

Under other circumstances, holders of certificates may wish to remain anonymous, but ensure that recipients of successive letters be aware that the letters came from the same sender, and to ensure integrity and/or confidentiality; this essentially preserves the anonymity currently provided by the ability to use arbitrary electronic mail names. One possible use may be to report a security problem anonymously. A certifying authority may issue *persona certificates* in which it explicitly is not vouching for the identity of the user. The final attribute of the issuing authority name in such a certificate is an Organizational Unit name with a value the distinguished string `PERSONA`. To continue the example, if Certificates, Inc. issued a persona certificate, the issuing authority name would be

/C=US/O=Certificates, Inc./OU=PERSONA/

In such a certificate, the subject name should be considered arbitrary and in no way reflective of the identity of the user to whom the certificate was issued.

Figure 9 summarizes the representations of the certifying authority with respect to subject identity and organizational affiliation for these types of certificates.

#### 4.2.4. Signing Certificates

A top-level certifying authority must provide for the certification of lower-level certifying authorities in such a way that private keys are not compromised. There are two scenarios possible, one in which the organization signs its own certificates and the other in which another entity signs certificates on behalf of the organization. In the former case, the organization will hold the key and use it to generate and sign the certificates it issues, whereas in the latter the other entity will hold

| *certificate* | *degree of affiliation* | *certainty of identity* |
|---|---|---|
| PERSONA | none | none |
| NOTARY | none | strong |
| GUEST | tenuous | strong |
| other | strong | strong |

Figure 9. How the different types of certificates bind subject (user) identity and affiliation with organizations named in the subject's Distinguished Name. The entries indicate the representations being made by the certifying authority in each case; so, for example, in a NOTARY certificate, the certifying authority has verified the user's identity but makes no claims about any organizational affiliations.

---

the private key and sign certificates for that organization. Under both scenarios, the organization is the certifying authority and must establish with the top-level certifying authority an appropriate pair of keys, called "organization keys," to sign certificates issued by the organization. The method for obtaining a certificate for oneself depends on where the organization private key is held.

#### 4.2.4.1. Another Entity Holds the Organization's Private Key

In this scenario, the organization might not have the equipment or expertise needed to protect adequately the certificate signature process, or its organizational private key. One option is to have another entity, such as the top-level certifying authority, hold the key and sign certificates on behalf of the organization. As the entity and the organization jointly issue the certificate (even though the organization is given as the issuer), this scenario is called the *co-issuer scenario* and the entity is called the *co-issuer.*

The specific legal and procedural requirements are up to the organization, the co-issuer, and the top-level certifying authority; they are expected to provide adequate protection both to the organization's private key and to the integrity and security of the signing of certificates. Basically, the user first generates a public and a private key. He must provide sufficient proof of identity and affiliation to the certifying authority to warrant its trust in the information it vouches for; normally, the organizational notary will determine when the proof is sufficient. The organizational notary will then forward this information along with the user's public key to the co-issuer using privacy-enhanced electronic mail to ensure integrity and authenticity. The co-issuer then constructs and signs the certificate on behalf of the certifying authority.

Notice that at no time does anyone other than the user need to know the user's private key; in particular, neither the organizational notary nor the co-issuer need it to generate the certificate. Further, no-one in the organization need ever know the private key used to sign the certificates it issues. Thus, the co-issuer must be trusted to protect that key adequately, and the top-level certification authority may employ procedural, technical, and legal means to encourage the provision of such protection.

#### 4.2.4.2. The Organization Holds Its Own Private Key

Some organizations may object to the loss of autonomy entailed by giving their private key to the co-issuer, or may find the co-issuer arrangement unsuitable for other reasons (*e.g.*, financial). Alternate arrangements are suitable provided they make certificate signing as reliable and as secure as the co-issuer scenario. Organizations signing their own certificates are their own *issuing authority*. Such organizations must ensure that the procedures and technology used provide as reliable and as secure a signature service as that offered by the co-issuer scenario, for example ensuring the organization's private key is adequately protected, that each certificate gets a unique serial number, and that the subject's Distinguished Name is subordinate to the issuer's Distinguished Name (except for `Notary` certificates). One example mechanism is discussed below in section 4.3.

### 4.2.5. Certificate Validation

Checking the validity of a certificate is straightforward. As noted above, each certificate contains an entity name, a public key, an issuer name, a digital signature created by the issuer based on its private key and the information in the certificate, and some other information. To validate that the certificate has not been altered, one need only obtain the issuer's certificate, extract the public key, and use that to decrypt the digital signature on the certificate in question. If this differs from the (locally computed) hash of the certificate in question, that certificate is bogus. The certificate of the issuer may be checked using the same procedure once the certificate of the top-level certifying authority (which will be widely publicized) is obtained.[62]

Note that the convention of having the top-level certification authority certify all other certification authorities in its domain means that within a domain, validation requires checking two certificates. If the original certificate came from another domain which is cross-certified by the local domain's top-level certification authority, then a third step is necessary: the certificate of the remote top-level certification authority must be checked by using the local top-level certification authority's certificate. So in the worst case, at most three certificates must be validated. Keeping the validation path so short was a primary consideration in defining the convention that the top-level certification authority certifies all certification authorities in its domain.

### 4.2.6. Certificate Revocation

Each certificate is issued for a limited period of time, and – more importantly – may be invalidated or compromised in a number of ways. For example, the entity name might change, or

---

62. [5], §3.4.2.

the private key associated with the public key in the certificate may be revealed. The international standard X.509 places the responsibility for maintaining time-stamped lists of revoked certificates, as well as revoked certificates representing certifying authorities, upon the certifying authority issuing the certificates. So the serial number of the compromised certificate is added to a revoked certificate list maintained by the certifying authority issuing the certificate. This list is signed by the private key of the certifying authority, is dated, and contains the date at which the next revoked certificate list will be issued. Although an addition to the format prescribed by X.509, this date enables a site to determine if the revoked list is the most recent one from that certifying authority. Note that this requires a new list to be issued at the stated time even if no certificates have been revoked; however, it also allows a site to determine if the list is out of date.[63]

Top-level certifying authorities are required to establish a database which maintains certificate revocation lists for all certification authorities in its domain, and that is accessible through electronic mail. Further, all certifying authorities must transmit their current list to the top-level certifying authority as well as to users and user agents within their domain.

Transmission of the lists may be done in a number of ways. The currently-defined mechanism is to use privacy-enhanced mail to propagate these certificate revocation lists. Such a letter contains three types of header fields: a `Certificate` field containing the certificate of the certifying authority issuing the list, any number of `Issuer-Certificate` fields, and a `CRL` field:

```
CRL: crl_encoded
```

containing the certificate revocation list `crl_encoded` encoded using the printable encoding defined in section 3.1.3. Future mechanisms may include the use of a USENET newsgroup.

### 4.3. Example of a Certifying Authority Hierarchy: RSA Data Security Inc.

There is no requirement that a single asymmetric cryptosystem be used to provide interchange keys; indeed, two such systems, one designed to provide confidentiality and the other to provide integrity and authenticity, could equally well be used. Such cryptosystems must be (believed to be) cryptographically strong; being (relatively) efficient to implement in either software or hardware, and expanding the encrypted information as little as possible, are also desirable properties (as the former means encrypting and/or authenticating takes little time, and the latter reduces the bandwidth used).

In addition to meeting all these requirements, the RSA cryptographic algorithm [35] is the primary algorithm recommended for use in international standards requiring (or recommending) use of a public key cryptosystem. So the initial asymmetric cryptographic algorithm defined in the proposal is the RSA algorithm.

63. [5], §3.3.3.2.

The basic requirements for the RSA algorithm in this context are that the modulus size vary between 508 and 1024 bits (or approximately $8.4\times10^{152}$ and $1.8\times10^{309}$),[64] and that the public exponent be either 3 or $2^{16}+1$ (the Fermat number $F_4$).[65] The former requirement is intended to make determining the private key acceptably hard, and the latter to allow the public key to be the modulus alone rather than the modulus and the public exponent; the algorithm identifier will indicate which of the two is being used. The public exponent $2^{16}+1$ was selected because it allows relatively efficient processing[66] and it is recommended by X.509, Annex G. The exponent 3 was chosen because exponentiation is even faster. Finally, if the message integrity check must be padded, it is to be padded on the left with the ASN.1 encoding of the identifier for the integrity checking algorithm and then with zeroes, and if the data encryption keys must be padded, they are to be padded with zeroes unless the message is addressed to multiple recipients and the public exponent is 3 [37]. In this specific case, a pseudorandom 64-bit quantity is to be generated for each recipient and as many copies as will fit are to be placed to the left of the data encryption key before encrypting.[67]

The RSA algorithm is covered within the United States by patents administered by RSA Data Security, Inc. (These patents do not apply outside the United States or to the United States government.) Currently RSA Data Security, Inc. plans to act as a top-level certifying authority, and plans to make available a free, source code implementation of privacy-enhanced electronic mail to purchasers of certificates.[68] The remainder of this section briefly describes their planned procedures for managing their certificate hierarchy, as a specific implementation of the more general discussion above.

---

64. The international standard [29] does not recommend a key length but suggests that "a value... of 512 bits be adopted initially, but subject to further study" (emphasis in original). Originally, these limits were 320 and 632 bits (about $2\times10^{97}$ and $2\times10^{191}$, respectively); these limits on the modulus size were chosen so that the software implementing the RSA cryptographic algorithms would be potentially exportable from the United States. However, other characteristics of software implementing these protocols also affects exportability, and given the success that Lenstra and Manasse have had in factoring numbers of around 100 digits [36], it was deemed prudent to increase the modulus size to a minimum of 512 bits.
65. [6], §3.1. The proposal also states that the public key is to consist of more than 100 digits ([5], §3.3.1); this means that if the exponent is 3, the modulus must be at least 100 digits. As noted in the previous footnote, the modulus should actually be much larger.
66. At most 17 multiplications are required to exponentiate to that power.
67. [37] discusses the theory behind this. The pseudorandom quantity concatenated with the exponent should contain at least half as many bits as the modulus, hence the change to [6], §3.1, which only required one 64-bit pseudorandom quantity for padding.
68. [5], §1.

### 4.3.1. Issuing Organizational Certificates

As a top-level certifying authority, RSA Data Security, Inc., plans to support both the co-issuer scenario (where the co-issuer is always the top-level certifying authority) and the scenario where the organization holds its own keys. Because the latter scenario requires the use of a device (the *certificate signature unit*) approved by RSA Data Security, Inc., to sign the certificates, it is called the CSU scenario.

The co-issuer scenario works as described in section 4.2.4.1. The co-issuer must be RSA Data Security, Inc., primarily to ensure an overall level of integrity by providing a uniform and a high degree of protection for the private keys. (Secondarily, it simplifies accounting and licensing issues.) The format of the organizational notary's message to RSA Data Security, Inc. will be defined by that top-level certifying authority. After the message is received, and RSA Data Security, Inc. authenticates the message, it issues a certificate signed with the private key of the organization.[69] This certificate will be valid for two years, at which point a new certificate must be acquired.

Under the co-issuer scenario, RSA Data Security Inc. will have overhead of extra personnel and equipment to run the software and to provide adequate management and protection for those keys and software. Hence it will charge a fee for each certificate signed to offset expenses as well as for the royalty.

An alternate arrangement is based on the procedure in section 4.2.4.2. It involves attaching to a workstation (or other computer) a tamperproof device known as a *certificate signature unit*.[70] Figure 10 summarizes the protocols that the certificate signature unit uses. A hardware random number generator seeds the generation of candidate RSA keys, each of which is subjected to various tests. When a pair of keys is selected, the private key is protected to very high standards; in fact, the key is stored unencrypted in tamperproof memory.

Because something catastrophic could destroy the certificate signature unit, thereby erasing the private key, a recovery mechanism is available. The organization has a secret key (the issuing authority secret key) stored on a removable key storage device; similarly, the manufacturer of the unit provides a secret key associated with that unit (the CSU secret key). The manufacturer retains a copy of this key. Once an organization's private key is selected, it is encrypted using the CSU secret key and the issuing authority's secret key, and the result is saved in a secure fashion (such as on a second removable key storage device; it may also be stored on a workstation, so it can be transmitted to the unit without a physical device being inserted). Should the certificate signature

69. [5], §3.3.3.

70. This mechanism for issuing, and paying for, certificates is not in [5], nor is it yet in place; BBN Communications is designing and building a prototype certificate signature unit.

unit be destroyed or stolen, the manufacturer can supply another unit with the same CSU secret key.

Once the certificate signature unit is obtained, and an organization's private and public keys generated, the public key must be transmitted to RSA Data Security, Inc., to enable validation of certificates issued by the organization. This is sent using authenticated, unencrypted privacy-enhanced mail, so the organization can be certain its private key is not being sent instead. It can also be used to account for the number of certificates issued.

In addition to the cost of acquiring the certificate signature unit itself, the organization is expected to pay a royalty for each certificate signed. Note that RSA Data Security, Inc., has a much smaller administrative overhead for certificates issued this way, so only the royalty is expected to be charged.

### 4.3.2. Organizations and Organizational Notaries

An organization registers by contacting RSA Data Security, Inc., and submitting an application. It must then keep RSA Data Security, Inc., apprised of any changes of administrative information and changes to the Distinguished Name used in the Issuer field of certificates it generates. If the latter occurs, it must re-register to maintain the association between the issuing authority Distinguished Name and the identity of the organization.

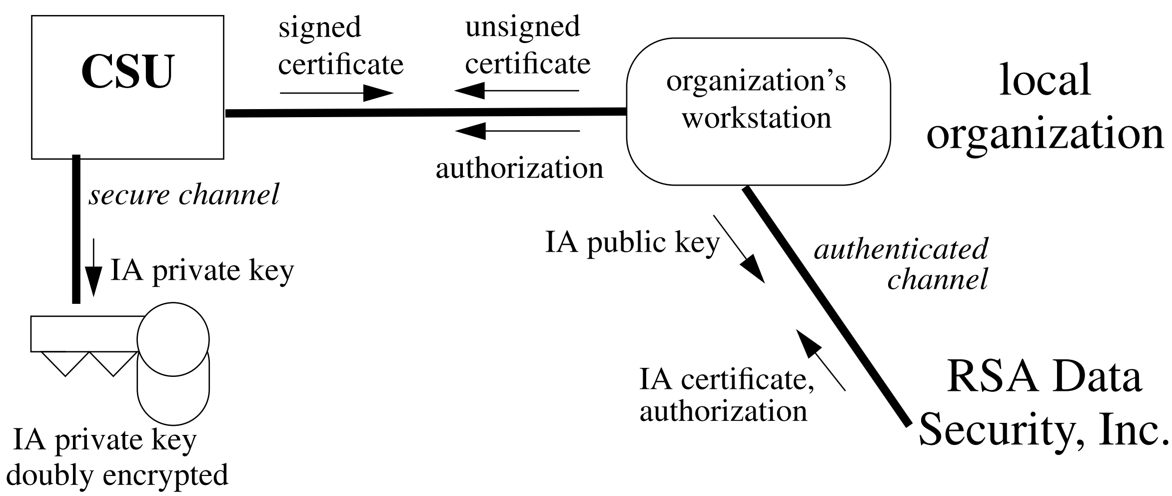


Figure 10. Protocols for the certificate signature unit. The issuing authority's private key is encrypted using a symmetric cryptosystem, once with the CSU's secret key and then with the IA's secret key, and is stored on a removable key storage device. The CSU receives unsigned certificates from a workstation (or other computer to which it is attached), checks their format, and signs them; this channel constitutes almost all of the traffic to and from the CSU. The organization obtains its certificate from RSA Data Security, Inc., as well as a message authorizing the CSU to sign signatures, which must then be sent to the CSU.

Each organization is to register one or more Organizational Notaries with RSA Data Security, Inc., as having authority to sign certificates for the organization. These Notaries are to be members of, and hold a position of trust in, that organization. The organization is free to impose additional requirements, and any changes or additions must be reported.

All certificate requests from that organization must go to one of the organization's Organizational Notaries; they must verify the authenticity of the information when applicable,[71] and either issue the certificate (under the CSU scenario) or forward the information to RSA Data Security, Inc., for it to issue the certificate (under the co-issuer scenario). They must also retain the information, and a copy of the certificate issued, for at least three years.

### 4.3.3. Certificate Revocation Lists

Certification authorities are to schedule regularly the issue of certificate revocation lists at least monthly; if the organization uses a certificate signature unit, it may issue its own list, but if it has a co-issuer, the organization must notify the co-issuer of the serial numbers to be placed on such a list, which the co-issuer will then issue. In the event of an emergency such as a compromise or an error in a released certificate revocation list, a new list may be issued whether or not it is scheduled. In any case, the certificate revocation list is to be retained by its issuer for at least three years, and if the co-issuer issues the list, the organization must keep the copy of the list sent to the co-issuer.

## 5. Implementation Requirements

Although the privacy-enhanced electronic mail proposals do not mandate details of implementation of the protocols, they do place certain constraints on what conforming implementations must do. The two most important restrictions are upon the processing of nested privacy-enhanced messages and error handling.

The proposals require conforming implementations to handle both recursively nested messages and serially nested messages; so, for example, a privacy-enhanced message (containing message headers, encapsulated message headers, and an encapsulated message body) may be made the encapsulated body of a second message and forwarded, and that message may itself be put into the encapsulated body of a third message and forwarded, and so on. Similarly, a set of privacy-enhanced messages may be placed one after the other and sent as the encapsulated body of another privacy-enhanced message. Precisely how the implementation handles such cases is up to the implementor, and one conforming prototype implementation simply requires the user to run a text ed-

71. This is not applicable for `PERSONA` certificates; however, there should be a way for a user with such a certificate to notify the Organizational Notary that the user's private key has been compromised.

itor on the message to extract each enclosed privacy-enhanced message and store it in its own file; each file may then be checked (and decrypted, if necessary). But some mechanism for doing this must be available and usable with the privacy-enhanced user agent.

More interesting are the requirements for handling errors. Syntax errors in encapsulated headers and bodies should be flagged, as should messages which yield an incorrect message integrity checksum. If the message is not of the type `MIC-CLEAR`, the latter case raises a very serious user interface issue: should the user be shown a message which may be useful even in its altered state? For example, if a character was deleted due to network transmission problems, the rest of the message may give the user enough indications to be able to reconstruct the sender and contact him for more information. On the other hand, if the entire message is forged, the user can place no reliance on it at all. The privacy-enhanced mail proposal strikes a balance between not showing the letter at all and showing the letter with a warning that the user may ignore (or not notice): it requires that the user be informed of the error and *then* be forced to ask that the relevant message be shown. This positive indication is the best technique to ensure the user is aware of the problem and accepts the risks of reading the message (and acting on it, if he does).

In any case, a user agent conforming to the requirements of the privacy-enhanced mail protocols must display the full distinguished name of the sender; if the message is encrypted, the full distinguished name of each recipient must also be shown. Further, if any certification path involves cross-certification of top-level certifying authorities, this must be made apparent to the user, as must the use of any persona certificates. The precise mechanism is up to the implementor.[72]

Finally, an error may occur because a privacy-enhanced letter is addressed incorrectly. In this case, most message transfer agents will return the letter to the originator. However, if the letter is encrypted, the sender could not read the contents, because he has none of the recipients' private keys. So, in order to enable the sender to process any encrypted message which is returned as undeliverable, all privacy-enhanced mail implementations must allow the sender to insert a `Key-Info` encapsulated header field after the `Originator-ID` encapsulated header field. This line may then be used to decrypt the message should it be returned.

## 6. A Symmetric Key Distribution Mechanism

Although the privacy-enhanced mail proposal recommends interchange keys be distributed using certificates, alternate key distribution methods may be used with it. In particular, given a trusted third party (server), symmetric cryptography may be used.

72. These details are not prescribed by either [4] or [5], but have since been adopted to ensure the user receives at least minimal information when something is amiss. As protocol specifications do not normally constrain implementations, these requirements are quite controversial.

The simplest such configuration [38] works by having each user establish a secret key known only to her and to the key distribution center. When one user wishes to send privacy-enhanced electronic mail to another, the sender obtains a data exchange key, encrypts the message in the normal manner, then encrypts the data exchange key and message integrity check using her interchange key, and transmits them to the trusted server. The server decrypts the two and encrypts them using its own key, along with the sender's identity. These are returned to the sender, who then inserts them into her message and sends the message. The recipient relays the information in the `Key-Info` and `Sender` fields to the trusted server, which decrypts the contents using its key and then encrypts them using the recipient's interchange key. When the recipient gets the result from the server, he can decrypt the message in the usual way.

The main problem with this scheme is that, in addition to trusting the user agents, the key distribution center must also be trusted; if it is not, the scheme is worthless. Since the proposal is intended as an internet standard, it cannot realistically expect an entity on the internet to trust servers *not* under its control. Hence, the key distribution proposal in support of privacy-enhanced electronic mail uses a public key cryptosystem and a mechanism that does not require trusting a single host.

Notice that trusting a certifying authority and trusting a server of the sort described here are very different. If the server is compromised, any of its subscribers can be impersonated and existing privacy-enhanced electronic letters read, as well as all letters sent after the compromise. However, if a certifying authority is compromised (by someone obtaining the secret keys it signs certificates with) the only damage is that new certificates can be forged, users impersonated, and electronic mail sent using those certificates can be read; electronic mail sent with certificates existing at the time of compromise is still protected, as are all letters sent before the compromise. In short, compromising a trusted server compromises everything; compromising a certifying authority does not compromise previously sent messages.

A second problem with trusted servers springs from the use of multiple servers in different domains introduces latencies in negotiating exchange keys. If someone at Dartmouth were to send a letter to someone at Purdue, the Dartmouth and Purdue servers would have to coordinate the selection of the interchange key. This would have to be done in a very timely fashion to prevent delays in the message transmission; this suggests replication of data on different servers, once again raising the issue of trusting a server not under local control. The alternative is to accept that encrypted mail might be delayed due to reasons beyond the control of the message transfer agents.

We should also note that this server does more than provide an authentication service: it provides a *digital signature service*. An authentication service confirms the identity of the user, and may provide credentials attesting to the authentication; it does not tie these to a message, thereby allowing repudiation. However, a digital signature service not only authenticates the user but also binds the user's identity to the message being sent, which allows non-repudiation if desired. An authentication tool would therefore be quite unsuitable for this scheme; in particular, the functionality of the authentication server *Kerberos* [39] would need to be radically expanded to make it suitable.

## 7. Forwarding Messages and Mailing Lists

Privacy-enhanced electronic mail can be forwarded, but if it is to be forwarded as authenticated by the forwarder and asymmetric interchange keys are used there is a subtlety. If the message is of type `MIC-ONLY` or `MIC-CLEAR`, the message integrity checksum in the `MIC-Info` header field is decrypted using the sender's public component, and re-encrypted using the recipient's private component; the message may then be forwarded; as an alternative, it can simply be placed within the body of another privacy-enhanced mail message. If the message is of type `ENCRYPTED`, however, it cannot be forwarded encrypted. The only possible approach (the recipient decrypts the data encryption key, uses that key and the originator's public component to decrypt the message integrity checksum, encrypts the message integrity checksum using his private component and the data encryption key, and then encrypts the data encryption key using the public component of the party to whom the message is to be forwarded) fails because a message may have only one originator; after the intermediate recipient processes the message, either he is the originator (and the original sender is lost), or there are two originators (which is illegal). Note also that in this case the encrypted message cannot simply be enclosed in another privacy-enhanced message, as the decryption requires the private key of the forwarder.

If symmetric interchange keys are used, the procedure is similar, except that the transformations involve agreed-upon secret keys, and in neither case can the message be forwarded simply by enclosing it in another privacy-enhanced mail message.

Sending a privacy-enhanced message to multiple recipients is straightforward and needs not be elaborated upon. However, the case when those multiple recipients are part of a mailing list (in which a letter sent to a single address is *exploded* or forwarded to multiple recipients) does.

In some cases, the membership of a mailing list may not be known or available to a sender for a variety of reasons. The host on which the alias is exploded may not be willing to reveal that information. The mailing list may itself contain mailing lists (for example, the mailing list *csnet-forum@relay.cs.net* has as an address *csnet-forum@dartmouth.edu*, which is itself a mailing list). In this case there must be an interchange key associated with the list; then all letters sent to the list are simply forwarded to each member of the list as described above.[73]

73. [4], §4.5.

If the sender can determine the membership of the mailing list, then a separate list interchange key is unnecessary; the sender can simply insert the destinations into the letter using multiple `Recipient-ID` and `Key-Info` encapsulated header fields. The message would be encrypted using the same data exchange key for all recipients and hence the encryption and integrity check need be done just once; however, the information in each `Key-Info` encapsulated header field would be encrypted using the associated recipient's public key.

If the certificate-based key distribution mechanism described above is used and the interchange keys are RSA keys with public exponent 3, note that even though there is a single recipient address there are multiple recipients. Hence the data exchange key should be padded with a 64-bit pseudorandom quantity, as discussed above.

## 8. Conclusion

The above two protocols satisfy the constraints and recommendations presented in the background section. Encapsulating the privacy-enhanced message renders it invisible to the message transfer agents, so no transport-level protocols need to be changed; the message may be processed by a special program and then included in the body of a conventional electronic mail message, so only that part of the user agent providing the privacy enhancements need be trusted with special security information, and conventional (non-enhanced) electronic mail is unaffected by the presence (or absence) of these enhancements. Integrity and authenticity are assured so long as the interchange keys are not compromised, and if the message is encrypted it will also be private. Finally, since the protocol for privacy-enhanced mail allows the use of any key management system, it is flexible enough to be used in a multitude of environments.

All this suggests an obvious architecture for implementing privacy-enhanced electronic mail. A special program to enhance messages runs above the user agent, taking as input a message and producing as output the transformed result. The user then includes this in a conventional mail message using any user agent desired. The recipient reverses the process by extracting the message from his mail message using his user agent, and then passing the encapsulated part to a second program which reverses the transformation and produces a local representation of the initial message.

The advent of personal computers and workstations has led many facilities to use central servers as mail hosts, so users can download their messages from their platform to the central host for transmission [40]. The above architecture fits into this scheme nicely; the privacy-enhancing software need reside only on the user's platform. This was no small consideration in the design of the protocol.

Both the X.411 Recommendation and MSP were designed for different requirements, and do not satisfy the same constraints as the privacy-enhanced mail protocols. Specifically, some of the security services described in the X.411 Recommendation require each message transfer agent to be able to parse the headers containing the security parameters; this would require altering existing message transfer agents. The privacy-enhanced mail protocols do not offer these services. By way of contrast, the Message Security Protocol is similar to privacy-enhanced mail, except that it uses the X.400 protocols rather than SMTP as its basis [41].

We must emphasize that the privacy-enhanced electronic mail protocol and the certificate-based key management protocol are distinct; one is free to adopt the first without using the second. Indeed, [4] specifically describes protocols to be used with key management schemes other than certificate-based schemes, and states that "the message processing procedures can also be used with symmetric key management."[74] Organizations that decide not to use the public key approach may substitute their own key management scheme; however, to be compatible with other implementations, all implementations of privacy-enhanced electronic mail should support the certificate-based approach.[75] This also means that the certificate-based key management protocols may be used in contexts other than privacy-enhanced mail [42]; since its infrastructure is similar to that of X.400, it can be used to transition to that, and related, standards.

More research in cryptography would aid in the maturation of this proposal. Specifically, one-way hash functions such as RSA-MD2 and RSA-MD4 that can be used to compute a message integrity check are very few; more are needed.

Finally, this proposal does not address issues in network and system security, such as the development of trusted software, routing controls, replays, and access controls. While all are important to the sending of electronic mail, they have much wider applications, and provide a fertile field for research and development.

***Acknowledgments.*** This paper presents a set of protocols developed during a series of meetings of the IAB's Privacy and Security Research Group. Thanks to David Balenson, Curt Barker, Jim Bidzos, Danny Cohen, Tom Daniel, Charles Fox, Morrie Gasser, Stephen Kent, John Laws, John Linn, Steve Lipner, Ralph Merkle, Dan Nessett, Mike Padlipsky, Ken Rossen, Rob Shirey, Miles Smid, Dave Solo, Steve Walker, and Steve Wilbur. Special thanks to Burton Kaliski, Jr., of RSA Data Security, Inc., for information on how RSA Data Security Inc. will handle the certification process, and to him, Ken Rossen, and Dave Solo on the certificate signature units, and to David Balenson for generating the sample message for symmetric interchange keys. Also, thanks to Donald Johnson, Burton Kaliski, Jr., John Linn, Evi Nemeth, Dan Nessett, J. Shallit, and especially David Balenson for their constructive comments on an earlier draft, to Joan Feigenbaum of AT&T Bell Laboratories for encouraging me to write this paper, and to the participants at the DIMACS Work-

74. [4], p. 10.
75. [4], §4.2.

shop on Distributed Computing and Cryptography, without whose probing questions (and spirited discussion) many of the parts of this paper would be a good deal less complete and more confusing.

## References


[1] M. Bishop, "Privacy-Enhanced Electronic Mail," *Distributed Computing and Cryptography*, American Mathematical Society, Providence, RI pp. 93-106 (1991).

[2] CCITT Recommendation X.411, *Message Handling Systems: Message Transfer System: Abstract Service Definition and Procedures* (1988).

[3] SDNS Protocol and Signaling Working Group, *SDNS Message Security Protocol*, SDN.701 Revision 1.5 (Aug. 1989)

[4] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I -- Message Encipherment and Authentication Procedures*, RFC-1113 (Aug. 1989).

[5] S. Kent and J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part II -- Certificate-Based Key Management*, RFC-1114 (Aug. 1989).

[6] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part III -- Algorithms, Modes, and Identifiers*, RFC-1115 (Aug. 1989).

[7] W. Barker, P. Cochrane, and M. Branstad, "Embedding Cryptography into a Trusted Mach System," *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, p. 379-383 (Dec. 1988).

[8] T. Casey and S. Wilbur, "Privacy Enhanced Electronic Mail," *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, pp. 16-21 (Dec. 1988).

[9] D. Dern, "The Trusted Mail System," *ConneXions – The Interoperability Report* **4**(2) p. 8 (Feb. 1990).

[10] J. Linn and S. Kent, "Electronic Mail Privacy Enhancement," *AIAA/ASIS/DODCI Second Aerospace Computer Security Conference*, pp. 40-44 (Dec. 1986).

[11] J. Linn and S. Kent, "Privacy for DARPA-Internet Mail," *Proceedings of the Twelfth National Computer Security Conference*, pp. 215-229 (Oct. 1989).

[12] CCITT Recommendation X.400, *Message Handling System Model* (1988).

[13] D. Crocker, *Standard for the Format of ARPA Internet Text Message*s, RFC-822 (Aug. 1982).

[14] J. Postel, *Simple Mail Transfer Protocol*, RFC-821 (Aug. 1982).

[15] D. Nowitz, and M. Lesk, *A Dial-Up Network of UNIX Systems*, document SMM:21 in *UNIX System Manager's Manual*, Computer Systems Research Group, University of California,

Berkeley, CA 94720 (Apr. 1986). Reprinted by the USENIX Association.

[16] V. Voydock and S. Kent, “Security Mechanisms in High-Level Network Protocols,” *Computing Surveys* **15**(2) pp. 135-171 (June 1983).

[17] D. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, MA (1982).

[18] *UNIX User’s Manual Reference Guide*, Computer Systems Research Group, University of California, Berkeley, CA 94720 (Apr. 1986). Reprinted by the USENIX Association.

[19] M. Rose and E. Stefferud, *Proposed Standard for Message Encapsulation*, RFC-934 (Jan. 1985).

[20] Federal Information Processing Standards Publication 113, *Computer Data Authentication* (May 1985).

[21] S. Akl, “On the Security of Compressed Encodings,” *Advances in Cryptology: Proceedings of Crypto 83*, Plenum Press, New York, NY (1984) pp. 209-230.

[22] C. Mitchell, D. Rush, and M. Walker, “A Remark on Hash Functions for Message Authentication,” *Computers and Security* **8**(1) pp. 55-58 (Feb. 1989).

[23] R. Rivest, *The MD4 Message Digest Algorithm*, RFC-1186 (Oct. 1990).

[24] Federal Information Processing Standards Publication 46, *Data Encryption Standard* (Jan. 1977).

[25] Federal Information Processing Standards Publication 81, *DES Modes of Operation* (Dec. 1980).

[26] CCITT Recommendation X.209, *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)* (1988)

[27] S. Kille, *Mapping Between X.400(1988) / ISO 10021 and RFC 822*, RFC-1148 (Mar. 1990).

[28] ANSI X9.17-1985, *American National Standard Financial Institution Key Management (Wholesale)*, American Bankers Association (Apr. 1985).

[29] CCITT Recommendation X.509, *The Directory - Authentication Framework* (1988).

[30] NIST Special Publication 500-183, *Stable Implementation Agreements for Open Systems Interconnection Protocols*, Version 4, Edition 1, Part 11 (Dec. 1990).

[31] CCITT Recommendation X.500, *The Directory - Overview of Concepts, Models, and Services* (1988).

[32] Federal Information Processing Standards Publication 146, *Government Open Systems Interconnection Profile (GOSIP)* (Apr. 1989).

[33] CCITT Recommendation X.208, *Specification of Abstract Syntax Notation One (ASN.1)*

(1988).

[34] ISO 3166, *Codes for the Representation of Names of Countries* (1987).

[35] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, **21**(2) pp. 120-126 (Feb. 1978).

[36] A. Lenstra and M. Manasse, "Factoring by Electronic Mail," to appear in *Advances in Cryptology – EUROCRYPT '89*, Springer-Verlag Berlin (1990).

[37] J. Moore, "Protocol Failures in Cryptosystems," *Proceedings of the IEEE*, **76**(5) pp. 597 (May 1988).

[38] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, **21**(12), pp. 993-999 (Dec. 1978).

[39] J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *USENIX Conference Proceedings*, pp. 191-202 (Winter 1988).

[40] D. Clark and M. Lambert, *PCMAIL: A Distributed Mail System for Personal Computers*, RFC-993 (Dec. 1986).

[41] R. Housley, "Electronic Messaging Security: A Comparison of Three Approaches," *Proceedings of the Fifth Annual Computer Security Applications Conference*, p. 29 (Dec. 1989).

[42] M. Bishop, "An Authentication Mechanism for USENET," *Winter 1991 USENIX Proceedings* pp. 281-287 (Jan. 1991).