S/MIME Implementation Guide

Interoperability Profile, Version 1

S/MIME Editor

Draft

Revised August 29, 1995

1. Overview

This paper provides implementation requirements and recommendations to ensure a base level of interoperability among S/MIME implementations.

The Internet electronic mail environment is comprised of a number of Message Transfer Agents and User Agents from various manufacturers on heterogeneous platforms. Interoperability is achieved through the adoption and implementation of message standards, i.e. RFC 822. Similarity of user experience has evolved to where a common set of basic features is present in most user agents.

Security services for Internet e-mail need to be interoperable not only at the message format layer but also in the support, treatment and presentation of security-critical information to ensure some commonality in the user experience.

The PKCS-MIME integration draft, "S/MIME Message Specification: PKCS Security Services for MIME," specifies a format for secure MIME messaging. The security portion of the message is defined by the PKCS #7 and #10. PKCS #7 is a flexible and extensible message format for representing the results of cryptographic operations on some data. Section 2 of this paper addresses some of the PKCS #7 algorithm and content options when using public-key cryptographic services. The PKCS #7 message format requires that public keys are bound to keyholder names within public key certificates. Section 3 of this paper provides implementation guidelines for the support of the distinguished name components within certificates. Section 4 of this paper deals with certificate processing. Section 5 deals with the generation of certification requests. Appendix A lists official Object Identifiers (OIDs) for the algorithms and attributes referenced in this paper.

Copyright © 1995 RSA Data Security, Inc.

Throughout this paper, there are separate requirements and recommendations made for the handling of incoming messages versus the creation of outgoing messages. In general, the often quoted, proverbial strategy is to "be liberal in what you receive and conservative in what you send". Most of the requirements are placed on the handling of incoming messages while the recommendations are mostly on the creation of outgoing messages. To rewrite the proverb somewhat, "you are required to be liberal in what you receive and it is recommended that you be conservative in what you send."

These requirements and recommendations are intended to form the basis of an S/MIME implementor's agreement for North American developers. Accordingly, it is anticipated that this will be revised in the future by standards efforts and market forces. Separate implementation profiles for other communities of interest (symmetric encryption, U.S. Government, foreign) are also anticipated.

2. PKCS #7 - Implementation Options

The PKCS #7 message format allows for a wide variety of options in content and algorithm support. This section puts forth a number of support requirements and recommendations in order to achieve a base level of interoperability among all S/MIME implementations. It should be noted that cryptographic compatibility can be achieved between PKCS #7 and Internet PEM (RFCs 1421-1424). This is addressed in detail throughout the PKCS #7 specification.

The following sections refer to specific sections in the PKCS #7 Version 1.5.

2.1 CertificateRevocationLists (PKCS #7: Section 6.1)

For incoming message, support for the Certificate Revocation List (CRL) format defined by Internet RFC 1422 is required. X.509 v2 CRL format is recommended. If CRLs are included in outgoing messages (see Section 4.5) then the CRL format defined by Internet RFC1422 is recommended.

2.2 ContentEncryptionAlgorithmIdentifier (PKCS #7: Section 6.2)

Due to U.S. Government export regulations, distribution of a S/MIME agent which supports a strong content encryption algorithm such as DES would not be freely exportable outside of North America. U.S. software manufacturers have been compelled to incorporate an "exportable" content encryption algorithm in order to create a widely exportable version of their product. S/MIME agents intended for domestic use (or use under State Department export license) can utilize stronger content encryption, however, in order to achieve interoperability such agents must support whatever exportable algorithm is incorporated in exportable S/MIME agents. Consequently, for incoming messages, support for RSA's RC2 algorithm in CBC mode

with effective key size of 40 bits is required. Where a separate domestic version is created, support for RC2 at higher key sizes, DES CBC, and DES EDE3-CBC for content encryption is recommended.

For outgoing messages, RC2 CBC at 40 bits is the recommended default. Stronger content encryption is strongly recommended where there is some mechanism to indicate that the intended recipient(s) can support it.

2.3 DigestAlgorithmIdentifier (PKCS #7: Section 6.3)

For incoming messages, support for RSA's MD2 and MD5 is required. For outgoing messages, MD5 is recommended.

2.4 DigestEncryptionAlgorithmIdentifier (PKCS #7: Section 6.4)

Support for rsaEncryption (defined by PKCS #1) is required. For incoming message, support for RSA key sizes from 512 bits to 1024 bits is required ***. Support for key sizes up to 2048 bits is recommended.

Outgoing messages are signed with a user's private key. For key generation recommendations, see Section 5.1.

*** This requirement is subject to a clarification on export rules from the US Government.

2.5 ExtendedCertificateOrCertificate (PKCS #7: Section 6.5)

The PKCS #7 message format supports a choice of certificate formats, X.509 and PKCS #6 Extended Certificates in public key content types (see PKCS #7 Section 6.5). The PKCS #6 format is not in widespread use. In addition, recent proposed revisions of X.509 certificates address much of the same functionality (flexibility) as was intended in the PKCS #6. Hence, the support and use of PKCS #6 extended certificates is not recommended.

For incoming message, support for X.509 v1 certificates is required. Support for X.509 v2 and v3 certificates is recommended.

Where certificates are included in outgoing messages (See section 2.6) X.509 v1 is recommended.

2.6 ExtendedCertificateAndCertificates (PKCS #7: Section 6.6)

PKCS #7 supports the inclusion of certificates in signed messages.

For incoming messages, an agent must be able to handle an arbitrary number of certificates of arbitrary relationship to the message sender and to each other in arbitrary order. Certificate processing is discussed later in Section 3.

For outgoing messages, it is recommended that a message originator include any certificates for the user's public key and associated issuer certificates. This increases the likelihood that the intended recipient can establish trust in the originator's public key. This is especially important when sending a message to recipients that may not have access to the sender's public key through any other means or when sending a signed message to a new recipient. The inclusion of certificates in outgoing messages can be omitted when e-mail is used within a group of correspondents that has established access to each other's certificates by some other means.

2.7 KeyEncryptionAlgorithmIdentifier (PKCS #7: Section 6.8)

Support for rsaEncryption (defined by PKCS #1) is required.

Incoming message keys are decrypted with the user's private key. For key generation recommendations, see Section 5.1.

For outgoing messages, support for RSA key sizes from 512 bits to 1024 bits is required.*** Support for key sizes up to 2048 bits is recommended.

*** This requirement is subject to a clarification on export rules from the US Government.

2.8 General Syntax (PKCS #7: Section 7)

The PKCS #7 defines six distinct content types: data, signedData, envelopedData, signedAndEnvelopedData, digestedData, and encryptedData.

For incoming messages, support for four types - data, signedData, signedAndEnvelopedData, and envelopedData - is required. Support for the remaining types; digestedData and encryptedData is recommended, however, this paper does not provide guidelines for interoperability using these content types.

For outgoing messages, the type of security service desired will dictate the data type to use. See following sections for discussion of various data types.

2.9 Data content type (PKCS #7: Section 8)

The data content type shall be used as the content within other content types to indicate the MIME message content which has had security services applied to it.

2.10 Signed-data content type (PKCS #7: Section 9)

This content type is used to apply a digital signature to a MIME message or, in a degenerate case where there is no signature information, to convey certificate and CRL information.

2.11 SignerInfo type (PKCS #7: Section 9.2)

The SignerInfo type allows the inclusion of unauthenticated and authenticated attributes to be included along with a signature.

For incoming messages, handling and displaying of zero or one instance of each of the following signed attributes is required.

For outgoing messages, generation of one instance of each of the following signed attributes is recommended. Where cryptographic compatibility with Internet PEM is desired, the message is required to have no attributes.

Attribute	Description
signingTime	PKCS #9
signingDescription	TBD

Note: Until there are trusted timestamping services, the time of signing will most likely be created by a message originator and therefore is only as trustworthy as the originator. The message description is intended to provide a short synopsis of the message which can be used to present a user with an additional confirmation step before committing to a cryptographic operation. In most cases, the replication of the Subject line from the header of a message should be sufficient and is recommended.

2.12 Enveloped-data content type (PKCS #7: Section 10)

This content type is used to apply privacy protection to a MIME message. Use of this service requires a sender to have access to a public key for each intended message recipient. This content type does not provide authentication.

2.13 Signed-and-enveloped-data content type (PKCS #7: Section 10)

This content type is used to apply a digital signature as well as privacy protection to a MIME message. Use of this service requires a sender to have access to a public key for each intended message recipient.

Copyright © 1995 RSA Data Security, Inc.

3. Distinguished Names in Certificates

3.1 Using Distinguished Names for e-mail

The format of an X.509 certificate includes fields for the subject name and issuer name. (The X.509 v1 certificate syntax is reproduced in Appendix A to PKCS #6). The subject name identifies the owner of a particular public key/private key pair while the issuer name is meant to identify the entity that "certified" the subject (i.e. signed the subject's certificate). The subject name and issuer name are defined by X.509 as Distinguished Names.

Distinguished Names are defined by a CCITT standard X.501. A Distinguished Name is broken into one or more Relative Distinguished Names. Each Relative Distinguished Name is comprised of one or more Attribute-Value Assertions. Each Attribute-Value Assertion consists of a Attribute Identifier and its corresponding value information, e.g. CountryName = US. Distinguished Names were intended to identify entities in the X.500 directory tree. Each Relative Distinguished Name can be thought of as a node in the tree which is described by some collection of Attribute-Value Assertions. The entire Distinguished Name is some collection of nodes in the tree that traverse a path from the root of the tree to some end node which represents a particular entity.

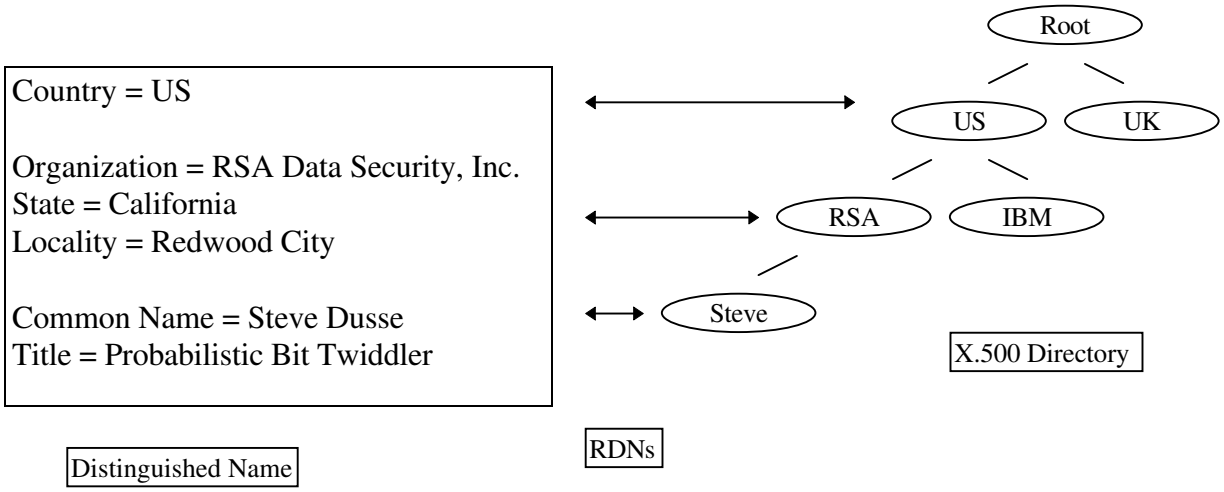

Figure 3.1 Example of a Distinguished Name

The (rather lofty) goal of the directory was to provide an infrastructure to uniquely name every communications entity everywhere (hence the Distinguished in Distinguished Name).

Adoption of a global X.500 directory infrastructure has been a little slower than expected. Consequently, the use of Distinguished Names in accordance with the X.500 directory is not very widespread. RFC 822 e-mail addresses are used almost

exclusively in the Internet environment to identify originators and recipients of messages. Internet e-mail addresses bear no resemblance to X.500 Distinguished Names (except, perhaps, that they are both hierarchical in nature). Some method is needed to map Internet e-mail addresses to entities that hold public keys. Some have suggested that the X.509 certificate format should be abandoned in favor of other binding mechanisms. (This is analogous to throwing out the baby with the bath water.) We recommend keeping the X.509 certificate and Distinguished Name mechanisms while tailoring the content of the naming information to suit the Internet e-mail environment (This is a lot like throwing out the baby and keeping the bath water.). At a minimum, either the Distinguished Name used to identify an Internet e-mail entity must include an RFC 822 e-mail address or some other mechanism must be implemented in the user agent to provide for Distinguished Name to e-mail address mapping. Where some Distinguished Name to e-mail address mapping mechanism is used, agents are required to provide authentication of the mapping or some method to force the user to confirm the mapping before employing the corresponding certificate.

NOTE: The creation and use of a Distinguished Name which includes only an RFC 822 e-mail address as the subject name in an X.509 certificate is encouraged. Such a construct, however, does not conform to the CCITT standards for Distinguished Names and should be used only as a transitional strategy. In many environments, the addition of classical Distinguished Name attributes to the RFC 822 address may be of value (e.g. inter-organization commerce).

3.2 Required Name Attributes

For incoming messages, support for parsing and display of zero, one, or more instances of each of the following set of name attributes within the Distinguished Names in certificates is required.

Inclusion of the RFC 822 e-mail address during Distinguished Name creation is highly recommended. Guidelines for the inclusion, omission, and ordering of the remaining name attributes during the creation of a distinguished name will most likely be dictated by the policies associated with the certification service which will certify the corresponding name and public key. See Section 5 for more discussion about certification requests.

Name Attribute	Description
CountryName	X.520
StateOrProvinceName	X.520
Locality	X.520
CommonName	X.520
Title	X.520
Organization	X.520
OrganizationalUnit	X.520
StreetAddress	X.520
Postal Code	X.520
Phone Number	X.520

Copyright © 1995 RSA Data Security, Inc.

4. Certificate Processing

4.1 Introduction

A public key certificate provides a mechanism to cryptographically demonstrate that the holder of the issuer key has signed and therefore bound the subject's public key to the subject's name. In most circumstances this indicates that the issuer has performed the service of actually verifying that the corresponding subject private key is actually held or controlled by the individual or entity identified by the subject public name. The issuer of a public key certificate is called a certifier or certification authority. The quality of this certification service (roughly corresponding to the veracity and trustworthiness of the binding) may vary greatly from one certifier to another. Different levels of service may be deemed appropriate for different applications. Certification authorities which provide a service within an organization will most likely do so in accordance with the needs and business practices of that organization. In general, certification authorities that provide services to others will publish some sort of policies and procedures which users can inspect to ascertain the level of service provided.

4.2 Trust Models

The goal of certification in a secure e-mail environment is to provide some level of trust in the public keys used between a message originator and recipient. To this end, it is necessary to create or identify some certification path between the originator and recipient. If, for instance, a certificate is created by the message recipient for the message originator's public key (and vice versa), then the message recipient will have a high level of trust in the public key used to verify the message originator's signature (and vice versa). This form is trust is referred to as "direct trust" as each correspondent directly establishes the identity and key of the other correspondent and certifies the other correspondent directly. This works very well for small communities (e.g. 2 users) but scales very poorly (for N correspondents there are roughly N^2 certificates to issue and manage).

For small workgroups or enterprises, a much better model is afforded by appointing or identifying a single individual or entity to act as a certification authority for the entire group. The certification authority assumes the responsibility to identify each correspondent in the group and vouch for the binding between the correspondent’s name and public key by creating a certificate signed by the authority's private key. All of the corespondents can then indirectly trust the keys of the other correspondents by identifying and trusting a single public key, that of the certification authority which can be used to verify the signatures on the certificates for the other correspondent’s public keys. (The procedures that the certification authority uses to certify the other

Copyright © 1995 RSA Data Security, Inc.

correspondents must be trusted also so it's always a good idea to select a trustworthy entity for your certification authority). While the role of a certification authority is logically distinct from that of a correspondent, the certification authority may, in fact be one of the actual correspondents. In this case, it may be helpful for the certification authority to maintain a separate (typically higher strength) keypair for the role of certification from the one that is used by the same individual for actual correspondence.

For e-mail environments that span multiple workgroups, multiple geographies or multiple enterprises, the use of a certification hierarchy is recommended. A certification hierarchy is created when a certification authority's public key is certified by another, higher level certification authority. The higher level certification authority may be an entity that is trusted by all parties or it may, in turn, be certified by an even higher level certification authority, and so on. The collection of certificates from some trusted high level authority to an end certificate (representing a user or e-mail correspondent) is called a "chain" of certificates. An example of an e-mail certification hierarchy is defined in the PEM-RFC 1422. In general, when an e-mail correspondent intends to correspond across geographical and organizational boundaries, consideration should be given to third-party certification authorities that act as inter-enterprise trust brokers. (Once again, the procedures that the certification authority uses to certify lower level certification authorities must be trusted also so care should be taken to research and select the appropriate entity or organization for your certification authority).

Support for indirect trust (workgroup trust) and hierarchical, centralized trust is recommended for secure e-mail. The use of direct or peer-to-peer trust is not recommended for secure e-mail integration as it is much more difficult to manage and scale. Where a small group of users (e.g. 2) wishes to communicate securely without the participation of an outside entity (third-party), a certification authority should be chosen among the group to manage a separate issuer key in an indirect trust model. This achieves an important goal, a single point of certification and correspondingly, a single point of revocation. It also allows for the use of secure e-mail by small groups with a smooth transition to certificate hierarchies.

4.3 Certificate Retrieval/Storage

A secure e-mail agent must provide some certificate retrieval mechanism in order to gain access to certificates for recipients of digital envelopes. There are a number of ways to implement certificate retrieval mechanisms. X.500 directory service is an excellent example of a certificate retrieval-only mechanism that is compatible with classic X.500 Distinguished Names. Another suggestion under consideration by the IETF is to provide certificate retrieval services as part of the existing Domain Name System (DNS). Until such mechanisms are widely used, their utility may be limited by the small number of correspondent's certificates that can be retrieved. At a minimum, a user agent could generate an e-mail message to an intended recipient requesting that recipient's certificate in a signed return message.

It is recommended that a secure e-mail agent also provide a mechanism to allow a user to “store” certificates for correspondents in such a way as to guarantee their later retrieval. In many environments, it may be desirable to link the certificate retrieval/storage mechanisms together in some sort of certificate database. In it’s simplest form, a certificate database would be local to a particular user and would function in a similar way as a “address book” that stores a user’s frequent e-mail correspondents. In this way, the certificate retrieval mechanism would be limited to the certificates that a user has stored (presumably from incoming messages). A comprehensive certificate retrieval/storage solution may combine two or more mechanisms to allow the greatest flexibility and utility to the user. For instance, a secure e-mail agent may resort to checking a centralized certificate retrieval mechanism for a certificate if it can not be found in a user’s local certificate storage/retrieval database.

4.4 Root Keys

It is important to provide some mechanism to "anchor" or "root" certificates for a particular user. A “root” key is one which is available (along with the associated name) to the user agent where the authentication mechanism is something other than a certificate signed by a higher level issuer . In a strictly hierarchical environment the collection of "root" public keys (those of the highest level certification authorities) may be embedded into the application in such a way as to make it difficult to alter without detection. This enables the user agent to utilize the trusted roots to verify certificates created by the highest level certification authorities for lower level certification authorities, the corresponding public keys can then be used to verify certificates for even lower level certification authorities and eventually for end-users. For indirect trust environments, the “root” keys are usually associated with some administrative entity such as the group leader or system administrator. As such, these keys are rarely known in advance and therefore difficult to “embed” into an application. In an indirect or mixed trust environment, the user agent must provide some mechanisms to allow the management (addition, deletion, etc.) of root keys on behalf of a user. These may be the keys of certification authorities that the user personally trusts, or more likely, the keys that a particular organization has established for certification. As such, the management of roots may be performed directly by the user or by some administrative entity. In either case, authentication of this information is critical to the security of the certificate information as the addition of a “rogue” root can create a significant weakness.

4.5 CRLs

A secure e-mail agent must have access to some certificate-revocation list retrieval mechanism in order to gain access to certificate-revocation information when validating certificate chains. It is recommended that a secure e-mail agent also provide a mechanism to allow a user to “store” incoming certificate-revocation information for

correspondents in such a way as to guarantee its later retrieval, however, it is always better to get the latest information from the CA than to get information stored away from incoming messages.

It is recommended that secure agents retrieve and utilize CRL information for every certificate that is verified as part of a certificate chain validation (see section 4.6), however, in many instances (such as off-line verification) access to the latest CRL information may be difficult or impossible. The use of CRL information, therefore, should be dictated by the value of the information that is protected. The value of the CRL information in a particular context is beyond the scope of this paper but may be governed by the policies associated with particular certificate hierarchies.

4.6 Certificate Chain Validation

In creating a user agent for secure e-mail, certificate, CRL and certificate chain validation must be as automated as possible while still acting in the best interests of the user. Certificate, CRL and chain validation must be performed when validating a correspondent's public key. This is necessary when a) verifying a signature from a correspondent and, b) creating a digital envelope with the correspondent as the intended recipient.

Certificates and CRLs are made available to the chain validation procedure in two ways: a) incoming mail messages, and b) certificate and CRL retrieval mechanisms. Certificates and CRLs in incoming messages are not required to be in any particular order nor are they required to be in any way related to the sender or recipient of the message (although in most cases they will be related to the sender). Incoming certificates and CRLs should be cached for use in chain validation and optionally stored for later use. This temporary certificate and CRL cache should be used to augment any other certificate and CRL retrieval mechanisms for chain validation on incoming signed messages.

A certificate chain validation always starts with a certificate associated with the intended correspondent and ends when there is some collection of certificates which links the correspondent's key to a "root" key that is known to the application or otherwise trusted by the user. In some cases, hierarchy associated with a particular root key may impose constraints on the depth of the certificate chain (number of certificates from the user to the root inclusive) as well as some constraints on the subordinance relationship between the issuer and the subject of certificates at a particular depth in the certificate chain. Subordinance is achieved when the issuer name is a subset of the subject name in a certificate.

It is required that some certificate depth constraint checking mechanism be supported on a per-root basis. It is recommended that some mechanism be supported to check the subordinance relationship between subject and issuer name at different levels in a certificate chain on a per-root basis.

The following procedure outlines a simplified certificate-chain checking procedure. More elaborate procedures may be appropriate where the certificate-chain environment is complex. Starting with the end-user certificate as the candidate certificate, a secure e-mail agent should:

1. Check to see if the issuer name matches one of the agent's trusted root certifier keys. If so, go to step 3. If not, proceed to step 2.

2. Attempt to retrieve a valid (non-expired, non-pending) certificate that has the candidate certificate's issuer name as subject name (higher level certificates for the issuer). If none, chain validation fails.

3. Attempt to use the certifier's public key (either within the higher level certificate or that of the root) to verify the signature on the candidate certificate. If the signature is not valid, chain validation fails. If the signature is valid, proceed to step 4.

4. Attempt to retrieve the latest, valid (non-expired, non-pending) Certificate-Revocation List for the certifier. If none, indicate failure and proceed to step 5. If found, check for candidate certificate's serial number among list of revoked certificates. If found, indicate revocation, chain validation fails. If not found, proceed to step 5.

5. If certifier's public key is not a trusted root, proceed to step 5. If certifier's public key is a trusted root, check for maximum chain depth constraint. If number of recursions exceeds maximum chain depth, chain validation fails. If not, chain validation is successful, quit (while you're ahead).

6. Using the new certifier's certificate, recurse to step 1.

When this certificate chain validation procedure is used for an incoming signed message, a user agent must present at a minimum the distinguished name information for the signer. It is recommended that some mechanism exist to present the name information for certifier's in the validated certificate chain and it is strongly recommended that some indication is made as to which "root" key the chain validation terminated with.

4.7 Certificate and CRL Signing Algorithms

Certificates and Certificate-Revocation Lists (CRLs) are signed by the certificate issuer. A secure agent must be capable of verifying the signatures on certificates and CRLs made with the md2WithRSA and md5WithRSA signature algorithms with key sizes from 512 bits to 1024 bits. ***

*** This requirement is subject to a clarification on export rules from the US Government.

5. Generating Keys and Certification Requests

5.1 Key Pair Generation

A secure e-mail user agent (or some related administrative utility or function) must be capable of generating an RSA key pair on behalf of the user. It is recommended that the key pair is generated from a good source of non-deterministic random input and protected in a secure fashion.

Due to U.S. Government export regulations, an exportable version of a user agent may be limited to generating 512 bit keys for users. A minimum key size for domestic users of 768 bits is recommended, 1024 for certification authorities. Where there are mechanisms to support the separation of keying material for signatures versus key management, exportable agents may be permitted to generate and use keys larger than 512 bits for signatures. ***

*** Note: Subject to clarification on export rules from the U.S. Government.

5.2 Binding Names and Keys

A secure e-mail user agent (or some related administrative utility) must be capable of generating a certification request given a user's public key and associated name information. In most cases, the user's public key/private key pair will be generated simultaneously, however, there are cases where the keying information may be generated by an external process (e.g. when a key pair is generated on a cryptographic token or by a "key recovery" service).

At no time should there exist multiple valid (i.e. non-expired, non-revoked) certificates for the same key pair bound to different Distinguished Names. Otherwise, a security flaw exists where an attacker can substitute one valid certificate for another in such a way that can not be detected by a message recipient. If a users wishes to change their name (or create an alternate name), the user agent should generate a new key pair. If the user wishes to reuse an existing key pair with a new or alternate name, the user should first have any valid certificates for the existing public key revoked.

In general, it is possible for a user to request certification for the same name and key from multiple certification authorities. This is not recommended, however, as the policies of different certification authorities may vary and this may create confusion among a user's correspondents as to which policy was used to certify the user.

In general, it is possible for a user to request certification for the same name with multiple key pairs from the same or different certification authorities. Although this may be somewhat confusing, this is acceptable for end-user certificates because the use of the Issuer Name/Serial Number pair in the PKCS #7 automatically disambiguates each user certificate. The use of the same name with different key pairs is not

recommended, however, for issuer keys, since certificate chain processing would be made more difficult by the task of having to "try" each different issuer key when validating a certificate created by that issuer.

5.3 Using PKCS #10 for Certification Requests

The PKCS-MIME integration draft, "PKCS Security Services for MIME" specifies the PKCS #10 message format for certification requests. PKCS #10 is a flexible and extensible message format for representing the results of cryptographic operations on some data. The choice of naming information is largely dictated by the policies and procedures associated with the intended certification service. Minimal requirements and recommendations for naming support are described in section 3.

In addition to key and naming information, the PKCS #10 format supports the inclusion of optional attributes, signed by the entity requesting certification. This allows for information to be conveyed in a certification request which may be useful to the request process, but not necessarily part of the Distinguished Name being certified.

For incoming requests, support for parsing and display of zero or one instance of each of the following set of certification-request attributes is required of certification-authorities.

Inclusion of the following attributes during the creation and submission of a certification-request will most likely be dictated by the policies associated with the certification service which will certify the corresponding name and public key.

Attribute	Description
Postal Address	X.520
Challenge Password	PKCS #9

A PKCS #10 request includes the public key of the entity to be certified and the request is signed by the entity requesting certification. Use of RSA keys is required.

For incoming requests, support for the identification of an RSA key with the rsa OID defined in X.509 and the rsaEncryption OID is required. In addition, support for the MD5 with RSA and MD2 with RSA signature algorithms is required of certification-authorities.

For the creation and submission of certification-requests, it is recommended that RSA keys are identified with the rsaEncryption OID and signed with the RSA with MD5 signature algorithm.

5.4 Fulfilling a Certification Request

It is recommended that certification authorities use the md5WithRSA signature algorithms when signing certificates.

5.5 Using PKCS #7 for Fulfilled Certificate Response

PKCS #7 supports a degenerate case of the SignedData content type where there are no signers on the content (and hence, the content value is "irrelevant"). This degenerate case is used to convey certificate and CRL information. Certification authorities are required to use this format for returning certificate information resulting from the successful fulfillment of a certification request. At a minimum, the inclusion of the actual subject certificate (corresponding to the information in the certification request) in the fulfilled certificate response is required. The inclusion of other certificates which link the issuer to higher level certification authorities and corresponding certificate-revocation lists is recommended. Unrelated certificates and revocation information is also acceptable.

Secure e-mail user agents are required to parse this degenerate PKCS #7 message type and handle the certificates and CRLs according to the requirements and recommendations in Section 4.

Appendix A - Object Identifiers

A.1 Content Encryption Algorithms

```
RC2-CBC OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) encryptionAlgorithm(3) 2}

DES-CBC OBJECT IDENTIFIER ::=
      {iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 7}

DES-EDE3-CBC OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) encryptionAlgorithm(3) 7}
```

A.2 Digest Algorithms

```
md2 OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 2}

md5 OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 5}
```

Copyright © 1995 RSA Data Security, Inc.

A.3 Asymmetric Encryption Algorithm

```
rsaEncryption OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1}

rsa OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) algorithm(8) encryptionAlgorithm(1) 1}
```

A.3 Signature Algorithms

```
md2WithRSAEncryption OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) 2}

md5WithRSAEncryption OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) 4}
```

A.4 Signed Attributes

```
signingTime OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-9(9) 5}

signingDescription OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-9(9) 13}
```

A.5 Name Attributes

```
e-mailAddress OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-9(9) 1}

CountryName OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 6}

StateOrProvinceName OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 8}

ocality OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 7}

CommonName OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 3}

Title OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 12}

Organization OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 10}

OrganizationalUnit OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 11}

StreetAddress OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 9}

Postal Code OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 17}
```

Copyright © 1995 RSA Data Security, Inc.

```
Phone Number OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 20}
```

A.6 Certification Request Attributes

```
Postal Address OBJECT IDENTIFIER ::=
      {joint-iso-ccitt(2) ds(5) attributeType(4) 16}

Challenge Password OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-9(9) 7}
```

Author's address

S/MIME Editor
RSA Data Security, Inc. (415) 595-8782
100 Marine Parkway, #500 (415) 595-1873 (fax)
Redwood City, CA 94065 USA smime-editor@rsa.com