Network Working Group
Request for Comments: 3183
Category: Experimental

T. Dean
W. Ottaway
QinetiQ

October 2001

Domain Security Services using S/MIME

Status of this Memo

Copyright Notice

Abstract

This document describes how the S/MIME (Secure/Multipurpose Internet Mail Extensions) protocol can be processed and generated by a number of components of a communication system, such as message transfer agents, guards and gateways to deliver security services. These services are collectively referred to as 'Domain Security Services'.

Acknowledgements

Significant comments were made by Luis Barriga, Greg Colla, Trevor Freeman, Russ Housley, Dave Kemp, Jim Schaad and Michael Zolotarev.

1. Introduction

The S/MIME [1] series of standards define a data encapsulation format for the provision of a number of security services including data integrity, confidentiality, and authentication. S/MIME is designed for use by messaging clients to deliver security services to distributed messaging applications.

The mechanisms described in this document are designed to solve a number of interoperability problems and technical limitations that arise when different security domains wish to communicate securely, for example when two domains use incompatible messaging technologies such as the X.400 series and SMTP/MIME, or when a single domain wishes to communicate securely with one of its members residing on an untrusted domain. The scenarios covered by this document are domain-to-domain, individual-to-domain and domain-to-individual

communications. This document is also applicable to organizations and enterprises that have internal PKIs which are not accessible by the outside world, but wish to interoperate securely using the S/MIME protocol.

There are many circumstances when it is not desirable or practical to provide end-to-end (desktop-to-desktop) security services, particularly between different security domains. An organization that is considering providing end-to-end security services will typically have to deal with some if not all of the following issues:

1) Heterogeneous message access methods: Users are accessing mail using mechanisms which re-format messages, such as using Web browsers. Message reformatting in the Message Store makes end-to-end encryption and signature validation impossible.

2) Message screening and audit: Server-based mechanisms such as searching for prohibited words or other content, virus scanning, and audit, are incompatible with end-to-end encryption.

3) PKI deployment issues: There may not be any certificate paths between two organizations. Or an organization may be sensitive about aspects of its PKI and unwilling to expose them to outside access. Also, full PKI deployment for all employees, may be expensive, not necessary or impractical for large organizations. For any of these reasons, direct end-to-end signature validation and encryption are impossible.

4) Heterogeneous message formats: One organization using X.400 series protocols wishes to communicate with another using SMTP. Message reformatting at gateways makes end-to-end encryption and signature validation impossible.

This document describes an approach to solving these problems by providing message security services at the level of a domain or an organization. This document specifies how these 'domain security services' can be provided using the S/MIME protocol. Domain security services may replace or complement mechanisms at the desktop. For example, a domain may decide to provide desktop-to-desktop signatures but domain-to-domain encryption services. Or it may allow desktop-to-desktop services for intra-domain use, but enforce domain-based services for communication with other domains.

Domain services can also be used by individual members of a corporation who are geographically remote and who wish to exchange encrypted and/or signed messages with their base.

Whether or not a domain based service is inherently better or worse than desktop based solutions is an open question. Some experts believe that only end-to-end solutions can be truly made secure, while others believe that the benefits offered by such things as content checking at domain boundaries offers considerable increase in practical security for many real systems. The additional service of allowing signature checking at several points on a communications path is also an extra benefit in many situations. This debate is outside the scope of this document. What is offered here is a set of tools that integrators can tailor in different ways to meet different needs in different circumstances.

Message transfer agents (MTAs), guards, firewalls and protocol translation gateways all provide domain security services. As with desktop based solutions, these components must be resilient against a wide variety of attacks intended to subvert the security services. Therefore, careful consideration should be given to security of these components, to make sure that their siting and configuration minimises the possibility of attack.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [2].

2. Overview of Domain Security Services

This section gives an informal overview of the security services that are provided by S/MIME between different security domains. These services are provided by a combination of mechanisms in the sender's and recipient's domains.

Later sections describe definitively how these services map onto elements of the S/MIME protocol.

The following security mechanisms are specified in this document:

1. Domain signature
2. Review signature
3. Additional attributes signature
4. Domain encryption and decryption

The signature types defined in this document are referred to as DOMSEC defined signatures.

The term 'security domain' as used in this document is defined as a collection of hardware and personnel operating under a single security authority and performing a common business function. Members of a security domain will of necessity share a high degree of mutual trust, due to their shared aims and objectives.

A security domain is typically protected from direct outside attack by physical measures and from indirect (electronic) attack by a combination of firewalls and guards at network boundaries. The interface between two security domains is termed a 'security boundary'. One example of a security domain is an organizational network ('Intranet').

2.1 Domain Signature

A domain signature is an S/MIME signature generated on behalf of a set of users in a domain. A domain signature can be used to authenticate information sent between domains or between a certain domain and one of its individuals, for example, when two 'Intranets' are connected using the Internet, or when an Intranet is connected to a remote user over the Internet. It can be used when two domains employ incompatible signature schemes internally or when there are no certification links between their PKIs. In both cases messages from the originator's domain are signed over the original message and signature (if present) using an algorithm, key, and certificate which can be processed by the recipient(s) or the recipient(s) domain. A domain signature is sometimes referred to as an "organizational signature".

2.2 Review Signature

A third party may review messages before they are forwarded to the final recipient(s) who may be in the same or a different security domain. Organizational policy and good security practice often require that messages be reviewed before they are released to external recipients. Having reviewed a message, an S/MIME signature is added to it - a review signature. An agent could check the review signature at the domain boundary, to ensure that only reviewed messages are released.

2.3 Additional Attributes Signature

A third party can add additional attributes to a signed message. An S/MIME signature is used for this purpose - an additional attributes signature. An example of an additional attribute is the 'Equivalent Label' attribute defined in ESS [3].

2.4 Domain Encryption and Decryption

Domain encryption is S/MIME encryption performed on behalf of a collection of users in a domain. Domain encryption can be used to protect information between domains, for example, when two 'Intranets' are connected using the Internet. It can also be used when end users do not have PKI/encryption capabilities at the desktop, or when two domains employ incompatible encryption schemes internally. In the latter case messages from the originator's domain are encrypted (or re-encrypted) using an algorithm, key, and certificate which can be decrypted by the recipient(s) or an entity in their domain. This scheme also applies to protecting information between a single domain and one of its members when both are connected using an untrusted network, e.g., the Internet.

3. Mapping of the Signature Services to the S/MIME Protocol

This section describes the S/MIME protocol elements that are used to provide the security services described above. ESS [3] introduces the concept of triple-wrapped messages that are first signed, then encrypted, then signed again. This document also uses this concept of triple-wrapping. In addition, this document also uses the concept of 'signature encapsulation'. 'Signature encapsulation' denotes a signed or unsigned message that is wrapped in a signature, this signature covering both the content and the first (inner) signature, if present.

Signature encapsulation MAY be performed on the inner and/or the outer signature of a triple-wrapped message.

For example, the originator signs a message which is then encapsulated with an 'additional attributes' signature. This is then encrypted. A reviewer then signs this encrypted data, which is then encapsulated by a domain signature.

There is a possibility that some policies will require signatures to be added in a specific order. By only allowing signatures to be added by encapsulation it is possible to determine the order in which the signatures have been added.

A DOMSEC defined signature MAY encapsulate a message in one of the following ways:

1) An unsigned message has an empty signature layer added to it (i.e., the message is wrapped in a signedData that has a signerInfos which contains no elements). This is to enable backward compatibility with S/MIME software that does not have a DOMSEC capability. Since the signerInfos will contain no signers

the eContentType, within the EncapsulatedContentInfo, MUST be id-data as described in CMS [5]. However, the eContent field will contain the unsigned message instead of being left empty as suggested in section 5.2 in CMS [5]. This is so that when the DOMSEC defined signature is added, as defined in method 2) below, the signature will cover the unsigned message.

2) Signature Encapsulation is used to wrap the original signed message with a DOMSEC defined signature. This is so that the DOMSEC defined signature covers the message and all the previously added signatures. Also, it is possible to determine that the DOMSEC defined signature was added after the signatures that are already there.

3.1 Naming Conventions and Signature Types

An entity receiving an S/MIME signed message would normally expect the signature to be that of the originator of the message. However, the message security services defined in this document require the recipient to be able to accept messages signed by other entities and/or the originator. When other entities sign the message the name in the certificate will not match the message sender's name. An S/MIME compliant implementation would normally flag a warning if there were a mismatch between the name in the certificate and the message sender's name. (This check prevents a number of types of masquerade attack.)

In the case of domain security services, this warning condition SHOULD be suppressed under certain circumstances. These circumstances are defined by a naming convention that specifies the form that the signers name SHOULD adhere to. Adherence to this naming convention avoids the problems of uncontrolled naming and the possible masquerade attacks that this would produce.

As an assistance to implementation, a signed attribute is defined to be included in the S/MIME signature - the 'signature type' attribute. On receiving a message containing this attribute, the naming convention checks are invoked.

Implementations conforming to this standard MUST support the naming convention for signature generation and verification. Implementations conforming to this standard MUST recognize the signature type attribute for signature verification. Implementations conforming to this standard MUST support the signature type attribute for signature generation.

3.1.1 Naming Conventions

The following naming conventions are specified for agents generating signatures specified in this document:

* For a domain signature, an agent generating this signature MUST be named 'domain-signing-authority'

* For a review signature, an agent generating this signature MUST be named 'review-authority'.

* For an additional attributes signature, an agent generating this signature MUST be named 'attribute-authority'.

This name shall appear as the 'common name (CN)' component of the subject field in the X.509 certificate. There MUST be only one CN component present. Additionally, if the certificate contains an RFC 822 address, this name shall appear in the end entity component of the address - on the left-hand side of the '@' symbol.

In the case of a domain signature, an additional naming rule is defined: the 'name mapping rule'. The name mapping rule states that for a domain signing authority, the domain part of its name MUST be the same as, or an ascendant of, the domain name of the message originator(s) that it is representing. The domain part is defined as follows:

* In the case of an X.500 distinguished subject name of an X.509 certificate, the domain part is the country, organization, organizational unit, state, and locality components of the distinguished name.

* In the case of an RFC 2247 distinguished name, the domain part is the domain components of the distinguished name.

* If the certificate contains an RFC 822 address, the domain part is defined to be the RFC 822 address component on the right-hand side of the '@' symbol.

For example, a domain signing authority acting on behalf of John Doe of the Acme corporation, whose distinguished name is 'cn=John Doe, ou=marketing,o=acme,c=us' and whose e-mail address is John.Doe@marketing.acme.com, could have a certificate containing a distinguished name of 'cn=domain-signing-authority,o=acme,c=us' and an RFC 822 address of 'domain-signing-authority@acme.com'. If John Doe has an RFC 2247

defined address of 'cn=John Doe,dc=marketing,dc=acme,dc=us' then an address of 'cn=domain-signing-authority,dc=acme,dc=us' could be used to represent the domain signing authority.

When the X.500 distinguished subject name has consecutive organizational units and/or localities it is important to understand the ordering of these values in order to determine if the domain part of the domain signature is an ascendant. In this case, when parsing the distinguished subject name from the most significant component (i.e., country, locality or organization) the parsed organizational unit or locality is deemed to be the ascendant of consecutive (unparsed) organizational units or localities.

When parsing an RFC 2247 subject name from the most significant component (i.e., the 'dc' entry that represents the country, locality or organization) the parsed 'dc' entry is deemed to be the ascendant of consecutive (unparsed) 'dc' entries.

For example, a domain signing authority acting on behalf of John Doe of the Acme corporation, whose distinguished name is 'cn=John Doe, ou=marketing,ou=defence,o=acme,c=us' and whose e-mail address is John.Doe@marketing.defence.acme.com, could have a certificate containing a distinguished name of 'cn=domain-signing-authority,ou=defence,o=acme,c=us' and an RFC 822 address of 'domain-signing-authority@defence.acme.com'. If John Doe has an RFC 2247 defined address of 'cn=John Doe,dc=marketing,dc=defense,dc=acme,dc=us' then the domain signing authority could have a distinguished name of 'cn=domain-signing-authority,dc=defence,dc=acme,dc=us'.

Any message received where the domain part of the domain signing agent's name does not match, or is not an ascendant of, the originator's domain name MUST be flagged.

This naming rule prevents agents from one organization masquerading as domain signing authorities on behalf of another. For the other types of signature defined in this document, no such named mapping rule is defined.

Implementations conforming to this standard MUST support this name mapping convention as a minimum. Implementations MAY choose to supplement this convention with other locally defined conventions. However, these MUST be agreed between sender and recipient domains prior to secure exchange of messages.

On verifying the signature, a receiving agent MUST ensure that the naming convention has been adhered to. Any message that violates the convention MUST be flagged.

3.1.2 Signature Type Attribute

An S/MIME signed attribute is used to indicate the type of signature. This should be used in conjunction with the naming conventions specified in the previous section. When an S/MIME signed message containing the signature type attribute is received it triggers the software to verify that the correct naming convention has been used.

The ASN.1 [4] notation of this attribute is: -

```
SignatureType ::= SEQUENCE OF OBJECT IDENTIFIER

id-sti  OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
            rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) 9 }

-- signature type identifier
```

If present, the SignatureType attribute MUST be a signed attribute, as defined in [5]. If the SignatureType attribute is absent and there are no further encapsulated signatures the recipient SHOULD assume that the signature is that of the message originator.

All of the signatures defined here are generated and processed as described in [5]. They are distinguished by the presence of the following values in the SignatureType signed attribute:

```
id-sti-domainSig OBJECT IDENTIFIER ::= { id-sti 2 }
-- domain signature.

id-sti-addAttribSig OBJECT IDENTIFIER ::= { id-sti 3 }
-- additional attributes signature.

id-sti-reviewSig OBJECT IDENTIFIER ::= { id-sti 4 }
-- review signature.
```

For completeness, an attribute type is also specified for an originator signature. However, this signature type is optional. It is defined as follows:

```
id-sti-originatorSig OBJECT IDENTIFIER ::= { id-sti 1 }
-- originator's signature.
```

All signature types, except the originator type, MUST encapsulate other signatures. Note a DOMSEC defined signature could be encapsulating an empty signature as defined in section 3.

A SignerInfo MUST NOT include multiple instances of SignatureType. A signed attribute representing a SignatureType MAY include multiple instances of different SignatureType values as an AttributeValue of attrValues [5], as long as the SignatureType 'additional attributes' is not present.

If there is more than one SignerInfo in a signerInfos (i.e., when different algorithms are used) then the SignatureType attribute in all the SignerInfos MUST contain the same content.

The following sections describe the conditions under which each of these types of signature may be generated, and how they are processed.

3.2 Domain Signature Generation and Verification

A 'domain signature' is a proxy signature generated on a user's behalf in the user's domain. The signature MUST adhere to the naming conventions in 3.1.1, including the name mapping convention. A 'domain signature' on a message authenticates the fact that the message has been released from that domain. Before signing, a process generating a 'domain signature' MUST first satisfy itself of the authenticity of the message originator. This is achieved by one of two methods. Either the 'originator's signature' is checked, if S/MIME signatures are used inside a domain. Or if not, some mechanism external to S/MIME is used, such as the physical address of the originating client or an authenticated IP link.

If the originator's authenticity is successfully verified by one of the above methods and all other signatures present are valid, including those that have been encrypted, a 'domain signature' can be added to a message.

If a 'domain signature' is added and the message is received by a Mail List Agent (MLA) there is a possibility that the 'domain signature' will be removed. To stop the 'domain signature' from being removed the steps in section 5 MUST be followed.

An entity generating a domain signature MUST do so using a certificate containing a subject name that follows the naming convention specified in 3.1.1.

If the originator's authenticity is not successfully verified or all the signatures present are not valid, a 'domain signature' MUST NOT be generated.

On reception, the 'domain signature' SHOULD be used to verify the authenticity of a message. A check MUST be made to ensure that both the naming convention and the name mapping convention have been used as specified in this standard.

A recipient can assume that successful verification of the domain signature also authenticates the message originator.

If there is an originator signature present, the name in that certificate SHOULD be used to identify the originator. This information can then be displayed to the recipient.

If there is no originator signature present, the only assumption that can be made is the domain the message originated from.

A domain signer can be assumed to have verified any signatures that it encapsulates. Therefore, it is not necessary to verify these signatures before treating the message as authentic. However, this standard does not preclude a recipient from attempting to verify any other signatures that are present.

The 'domain signature' is indicated by the presence of the value id-sti-domainSig in a 'signature type' signed attribute.

There MAY be one or more 'domain signature' signatures in an S/MIME encoding.

3.3 Additional Attributes Signature Generation and Verification

The 'additional attributes' signature type indicates that the SignerInfo contains additional attributes that are associated with the message.

All attributes in the applicable SignerInfo MUST be treated as additional attributes. Successful verification of an 'additional attributes' signature means only that the attributes are authentically bound to the message. A recipient MUST NOT assume that its successful verification also authenticates the message originator.

An entity generating an 'additional attributes' signature MUST do so using a certificate containing a subject name that follows the naming convention specified in 3.1.1. On reception, a check MUST be made to ensure that the naming convention has been used.

A signer MAY include any of the attributes listed in [3] or in this document when generating an 'additional attributes' signature. The following attributes have a special meaning, when present in an 'additional attributes' signature:

1) Equivalent Label: label values in this attribute are to be treated as equivalent to the security label contained in an encapsulated SignerInfo, if present.

2) Security Label: the label value indicates the aggregate sensitivity of the inner message content plus any encapsulated signedData and envelopedData containers. The label on the original data is indicated by the value in the originator's signature, if present.

An 'additional attributes' signature is indicated by the presence of the value id-sti-addAttribSig in a 'signature type' signed attribute. Other Object Identifiers MUST NOT be included in the sequence of OIDs if this value is present.

There MAY be multiple 'additional attributes' signatures in an S/MIME encoding.

3.4 Review Signature Generation and Verification

The review signature indicates that the signer has reviewed the message. Successful verification of a review signature means only that the signer has approved the message for onward transmission to the recipient(s). When the recipient is in another domain, a device on a domain boundary such as a Mail Guard or firewall may be configured to check review signatures. A recipient MUST NOT assume that its successful verification also authenticates the message originator.

An entity generating a signed review signature MUST do so using a certificate containing a subject name that follows the naming convention specified in 3.1.1. On reception, a check MUST be made to ensure that the naming convention has been used.

A review signature is indicated by the presence of the value id-sti-reviewSig in a 'signature type' signed attribute.

There MAY be multiple review signatures in an S/MIME encoding.

3.5 Originator Signature

The 'originator signature' is used to indicate that the signer is the originator of the message and its contents. It is included in this document for completeness only. An originator signature is indicated either by the absence of the signature type attribute, or by the presence of the value id-sti-originatorSig in a 'signature type' signed attribute.

4. Encryption and Decryption

Message encryption may be performed by a third party on behalf of a set of originators in a domain. This is referred to as domain encryption. Message decryption may be performed by a third party on behalf of a set of recipients in a domain. This is referred to as domain decryption. The third party that performs these processes is referred to in this section as a "Domain Confidentiality Authority" (DCA). Both of these processes are described in this section.

Messages may be encrypted for decryption by the final recipient and/or by a DCA in the recipient's domain. The message may also be encrypted for decryption by a DCA in the originator's domain (e.g., for content analysis, audit, key word scanning, etc.). The choice of which of these is actually performed is a system specific issue that depends on system security policy. It is therefore outside the scope of this document. These processes of encryption and decryption processes are shown in the following table.

	Recipient Decryption	Domain Decryption
Originator Encryption	Case(a)	Case(b)
Domain Encryption	Case(c)	Case(d)

Case (a), encryption of messages by the originator for decryption by the final recipient(s), is described in CMS [5]. In cases (c) and (d), encryption is performed not by the originator but by the DCA in the originator's domain. In cases (b) and (d), decryption is performed not by the recipient(s) but by the DCA in the recipient's domain.

A client implementation that conforms to this standard MUST support case (b) for transmission, case (c) for reception and case (a) for transmission and reception.

A DCA implementation that conforms to this standard MUST support cases (c) and (d), for transmission, and cases (b) and (d) for reception. In cases (c) and (d) the 'domain signature' SHOULD be applied before the encryption. In cases (b) and (d) the message SHOULD be decrypted before the originators 'domain signature' is obtained and verified.

The process of encryption and decryption is documented in CMS [5]. The only additional requirement introduced by domain encryption and decryption is for greater flexibility in the management of keys, as described in the following subsections. As with signatures, a naming convention and name mapping convention are used to locate the correct public key.

The mechanisms described below are applicable both to key agreement and key transport systems, as documented in CMS [5]. The phrase 'encryption key' is used as a collective term to cover the key management keys used by both techniques.

The mechanisms below are also applicable to individual roving users who wish to encrypt messages that are sent back to base.

4.1 Domain Confidentiality Naming Conventions

A DCA MUST be named 'domain-confidentiality-authority'. This name MUST appear in the 'common name(CN)' component of the subject field in the X.509 certificate. Additionally, if the certificate contains an RFC 822 address, this name MUST appear in the end entity part of the address, i.e., on the left-hand side of the '@' symbol.

Along with this naming convention, an additional naming rule is defined: the 'name mapping rule'. The name mapping rule states that for a DCA, the domain part of its name MUST be the same as, or an ascendant of (as defined in section 3.1.1), the domain name of the set of entities that it represents. The domain part is defined as follows:

* In the case of an X.500 distinguished name of an X.509 certificate, the domain part is the country, organization, organizational unit, state, and locality components of the distinguished name.

* In the case of an RFC 2247 distinguished name, the domain part is the domain components of the distinguished name.

* If the certificate contains an RFC 822 address, the domain part is defined to be the RFC 822 address part on the right-hand side of the '@' symbol.

For example, a DCA acting on behalf of John Doe of the Acme corporation, whose distinguished name is 'cn=John Doe,ou=marketing, o=acme,c=us' and whose e-mail address is John.Doe@marketing.acme.com, could have a certificate containing a distinguished name of 'cn=domain-confidentiality-authority,o=acme,c=us' and an e-mail address of 'domain-confidentiality-authority@acme.com'. If John Doe has an RFC 2247 defined address of 'cn=John Doe,dc=marketing, dc=defense,dc=acme,dc=us' then the domain signing authority could have a distinguished name of 'cn=domain-signing-authority,dc=defence,dc=acme,dc=us'. The key associated with this certificate would be used for encrypting messages for John Doe.

Any message received where the domain part of the domain encrypting agents name does not match, or is not an ascendant of, the domain name of the entities it represents MUST be flagged.

This naming rule prevents messages being encrypted for the wrong domain decryption agent.

Implementations conforming to this standard MUST support this name mapping convention as a minimum. Implementations may choose to supplement this convention with other locally defined conventions. However, these MUST be agreed between sender and recipient domains prior to sending any messages.

4.2 Key Management for DCA Encryption

At the sender's domain, DCA encryption is achieved using the recipient DCA's certificate or the end recipient's certificate. For this, the encrypting process must be able to correctly locate the certificate for the corresponding DCA in the recipient's domain or the one corresponding to the end recipient. Having located the correct certificate, the encryption process is then performed and additional information required for decryption is conveyed to the recipient in the recipientInfo field as specified in CMS [5]. A DCA encryption agent MUST be named according to the naming convention specified in section 4.1. This is so that the corresponding certificate can be found.

No specific method for locating the certificate to the corresponding DCA in the recipient's domain or the one corresponding to the end recipient is mandated in this document. An implementation may choose to access a local certificate store to locate the correct certificate. Alternatively, a X.500 or LDAP directory may be used in one of the following ways:

1. The directory may store the DCA certificate in the recipient's directory entry. When the user certificate attribute is requested, this certificate is returned.

2. The encrypting agent maps the recipient's name to the DCA name in the manner specified in 4.1. The user certificate attribute associated with this directory entry is then obtained.

This document does not mandate either of these processes. Whichever one is used, the name mapping conventions must be adhered to, in order to maintain confidentiality.

Having located the correct certificate, the encryption process is then performed. A recipientInfo for the DCA or end recipient is then generated, as described in CMS [5].

DCA encryption may be performed for decryption by the end recipient and/or by a DCA. End recipient decryption is described in CMS [5]. DCA decryption is described in section 4.3.

4.3 Key Management for DCA Decryption

DCA decryption uses a private-key belonging to the DCA and the necessary information conveyed in the DCA's recipientInfo field.

It should be noted that domain decryption can be performed on messages encrypted by the originator and/or by a DCA in the originator's domain. In the first case, the encryption process is described in CMS [5]; in the second case, the encryption process is described in 4.2.

5. Applying a Domain Signature when Mail List Agents are Present.

It is possible that a message leaving a DOMSEC domain may encounter a Mail List Agent (MLA) before it reaches the final recipient. There is a possibility that this would result in the 'domain signature' being stripped off the message. We do not want a MLA to remove the 'domain signature'. Therefore, the 'domain signature' must be applied to the message in such a way that will prevent a MLA from removing it.

A MLA will search a message for the "outer" signedData layer, as defined in ESS [3] section 4.2, and strip off all signedData layers that encapsulate this "outer" signedData layer. Where this "outer" signedData layer is found will depend on whether the message contains a mlExpansionHistory attribute or an envelopedData layer.

There is a possibility that a message leaving a DOMSEC domain has already been processed by a MLA, in which case a 'mlExpansionHistory' attribute will be present within the message.

There is a possibility that the message will contain an envelopedData layer. This will be the case when the message has been encrypted within the domain for the domain's "Domain Confidentiality Authority", see section 4.0, and, possibly, the final recipient.

How the 'domain signature' is applied will depend on what is already present within the message. Before the 'domain signature' can be applied the message MUST be searched for the "outer" signedData layer, this search is complete when one of the following is found: -

- The "outer" signedData layer that includes an mlExpansionHistory attribute or encapsulates an envelopedData object.
- An envelopedData layer.
- The original content (that is, a layer that is neither envelopedData nor signedData).

If a signedData layer containing a mlExpansionHistory attribute has been found then: -

1) Strip off the signedData layer (after remembering the included signedAttributes).

2) Search the rest of the message until an envelopedData layer or the original content is found.

3) a) If an envelopedData layer has been found then: -

   - Strip off all the signedData layers down to the envelopedData layer.
   - Locate the RecipientInfo for the local DCA and use the information it contains to obtain the message key.
   - Decrypt the encryptedContent using the message key.
   - Encapsulate the decrypted message with a 'domain signature'
   - If local policy requires the message to be encrypted using S/MIME encryption before leaving the domain then encapsulate the 'domain signature' with an envelopedData layer containing RecipientInfo structures for each of the recipients and an originatorInfo value built from information describing this DCA.

If local policy does not require the message to be encrypted using S/MIME encryption but there is an envelopedData at a lower level within the message then the 'domain signature' MUST be encapsulated by an envelopedData as described above.

An example when it may not be local policy to require S/MIME encryption is when there is a link crypto present.

b) If an envelopedData layer has not been found then: -

- Encapsulate the new message with a 'domain signature'.

4) Encapsulate the new message in a signedData layer, adding the signedAttributes from the signedData layer that contained the mlExpansionHistory attribute.

If no signedData layer containing a mlExpansionHistory attribute has been found but an envelopedData has been found then: -

1) Strip off all the signedData layers down to the envelopedData layer.
2) Locate the RecipientInfo for the local DCA and use the information it contains to obtain the message key.
3) Decrypt the encryptedContent using the message key.
4) Encapsulate the decrypted message with a 'domain signature'
5) If local policy requires the message to be encrypted before leaving the domain then encapsulate the 'domain signature' with an envelopedData layer containing RecipientInfo structures for each of the recipients and an originatorInfo value built from information describing this DCA.

   If local policy does not require the message to be encrypted using S/MIME encryption but there is an envelopedData at a lower level within the message then the 'domain signature' MUST be encapsulated by an envelopedData as described above.

If no signedData layer containing a mlExpansionHistory attribute has been found and no envelopedData has been found then: -

1) Encapsulate the message in a 'domain signature'.

5.1 Examples of Rule Processing

The following examples help explain the above rules. All of the signedData objects are valid and none of them are a domain signature. If a signedData object was a domain signature then it would not be necessary to validate any further signedData objects.

1) A message (S1 (Original Content)) (where S = signedData) in which the signedData does not include an mlExpansionHistory attribute is to have a 'domain signature' applied. The signedData, S1, is verified. No "outer" signedData is found, after searching for one as defined above, since the original content is found, nor is an envelopedData or a mlExpansionHistory attribute found. A new signedData layer, S2, is created that contains a 'domain signature', resulting in the following message sent out of the domain (S2 (S1 (Original Content))).

2) A message (S3 (S2 (S1 (Original Content))) in which none of the signedData layers includes an mlExpansionHistory attribute is to have a 'domain signature' applied. The signedData objects S1, S2 and S3 are verified. There is not an original, "outer" signedData layer since the original content is found, nor is an envelopedData or a mlExpansionHistory attribute found. A new signedData layer, S4, is created that contains a 'domain signature', resulting in the following message sent out of the domain (S4 (S3 (S2 (S1 (Original Content))).

3) A message (E1 (S1 (Original Content))) (where E = envelopedData) in which S1 does not include a mlExpansionHistory attribute is to have a 'domain signature' applied. There is not an original, received "outer" signedData layer since the envelopedData, E1, is found at the outer layer. The encryptedContent is decrypted. The signedData, S1, is verified. The decrypted content is wrapped in a new signedData layer, S2, which contains a 'domain signature'. If local policy requires the message to be encrypted, using S/MIME encryption, before it leaves the domain then this new message is wrapped in an envelopedData layer, E2, resulting in the following message sent out of the domain (E2 (S2 (S1 (Original Content)))), else the message is not wrapped in an envelopedData layer resulting in the following message (S2 (S1 (Original Content))) being sent.

4) A message (S2 (E1 (S1 (Original Content)))) in which S2 includes a mlExpansionHistory attribute is to have a 'domain signature' applied. The signedData object S2 is verified. The mlExpansionHistory attribute is found in S2, so S2 is the "outer" signedData. The signed attributes in S2 are remembered for later inclusion in the new outer signedData that is applied to the message. S2 is stripped off and the message is decrypted. The signedData object S1 is verified. The decrypted message is wrapped in a signedData layer, S3, which contains a 'domain signature'. If local policy requires the message to be encrypted, using S/MIME encryption, before it leaves the domain then this new message is wrapped in an envelopedData layer, E2. A new signedData layer, S4, is then wrapped around the envelopedData,

E2, resulting in the following message sent out of the domain (S4 (E2 (S3 (S1 (Original Content))))). If local policy does not require the message to be encrypted, using S/MIME encryption, before it leaves the domain then the message is not wrapped in an envelopedData layer but is wrapped in a new signedData layer, S4, resulting in the following message sent out of the domain (S4 (S3 (S1 (Original Content). The signedData S4, in both cases, contains the signed attributes from S2.

5) A message (S3 (S2 (E1 (S1 (Original Content))))) in which none of the signedData layers include a mlExpansionHistory attribute is to have a 'domain signature' applied. The signedData objects S3 and S2 are verified. When the envelopedData E1 is found the signedData objects S3 and S2 are stripped off. The encryptedContent is decrypted. The signedData object S1 is verified. The decrypted content is wrapped in a new signedData layer, S4, which contains a 'domain signature'. If local policy requires the message to be encrypted, using S/MIME encryption, before it leaves the domain then this new message is wrapped in an envelopedData layer, E2, resulting in the following message sent out of the domain (E2 (S4 (S1 (Original Content)))), else the message is not wrapped in an envelopedData layer resulting in the following message (S4 (S1 (Original Content))) being sent.

6) A message (S3 (S2 (E1 (S1 (Original Content))))) in which S3 includes a mlExpansionHistory attribute is to have a 'domain signature' applied. The signedData objects S3 and S2 are verified. The mlExpansionHistory attribute is found in S3, so S3 is the "outer" signedData. The signed attributes in S3 are remembered for later inclusion in the new outer signedData that is applied to the message. The signedData object S3 is stripped off. When the envelopedData layer, E1, is found the signedData object S2 is stripped off. The encryptedContent is decrypted. The signedData object S1 is verified. The decrypted content is wrapped in a new signedData layer, S4, which contains a 'domain signature'. If local policy requires the message to be encrypted, using S/MIME encryption, before it leaves the domain then this new message is wrapped in an envelopedData layer, E2. A new signedData layer, S5, is then wrapped around the envelopedData, E2, resulting in the following message sent out of the domain (S5 (E2 (S4 (S1 (Original Content))))). If local policy does not require the message to be encrypted, using S/MIME encryption, before it leaves the domain then the message is not wrapped in an envelopedData layer but is wrapped in a new signedData layer, S5, resulting in the following message sent out of the domain (S5 (S4 (S1 (Original Content). The signedData S5, in both cases, contains the signed attributes from S3.

7) A message (S3 (E2 (S2 (E1 (S1 (Original Content)))))) in which S3 does not include a mlExpansionHistory attribute is to have a 'domain signature' applied. The signedData object S3 is verified. When the envelopedData E2 is found the signedData object S3 is stripped off. The encryptedContent is decrypted. The signedData object S2 is verified, the envelopedData E1 is decrypted and the signedData object S1 is verified. The signedData object S2 is wrapped in a new signedData layer S4, which contains a 'domain signature'. Since there is an envelopedData E1 lower down in the message, the new message is wrapped in an envelopedData layer, E3, resulting in the following message sent out of the domain (E3 (S4 (S2 (E1 (S1 (Original Content)))))).

6. Security Considerations

This specification relies on the existence of several well known names, such as domain-confidentiality-authority. Organizations must take care with these names, even if they do not support DOMSEC, so that certificates issued in these names are only issued to legitimate entities. If this is not true then an individual could get a certificate associated with domain-confidentiality-authority@acme.com and as a result might be able to read messages the a DOMSEC client intended for others.

Implementations MUST protect all private keys. Compromise of the signer's private key permits masquerade.

Similarly, compromise of the content-encryption key may result in disclosure of the encrypted content.

Compromise of key material is regarded as an even more serious issue for domain security services than for an S/MIME client. This is because compromise of the private key may in turn compromise the security of a whole domain. Therefore, great care should be used when considering its protection.

Domain encryption alone is not secure and should be used in conjunction with a domain signature to avoid a masquerade attack, where an attacker that has obtained a DCA certificate can fake a message to that domain pretending to be another domain.

When an encrypted DOMSEC message is sent to an end user in such a way that the message is decrypted by the end users DCA the message will be in plain text and therefore confidentiality could be compromised.

If the recipient's DCA is compromised then the recipient can not guarantee the integrity of the message. Furthermore, even if the recipient's DCA correctly verifies a message's signatures, then a message could be undetectably modified, when there are no signatures on a message that the recipient can verify.

7. DOMSEC ASN.1 Module

```
DOMSECSyntax
 { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) domsec(10) }

 DEFINITIONS IMPLICIT TAGS ::=
 BEGIN

 -- EXPORTS All
 -- The types and values defined in this module are exported for
 -- use in the other ASN.1 modules.  Other applications may use
 -- them for their own purposes.

 SignatureType ::= SEQUENCE OF OBJECT IDENTIFIER

 id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
          us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 16 }

 id-sti  OBJECT IDENTIFIER ::= { id-smime 9 }   -- signature type
 identifier

 -- Signature Type Identifiers

 id-sti-originatorSig       OBJECT IDENTIFIER ::= { id-sti 1 }
 id-sti-domainSig           OBJECT IDENTIFIER ::= { id-sti 2 }
 id-sti-addAttribSig        OBJECT IDENTIFIER ::= { id-sti 3 }
 id-sti-reviewSig           OBJECT IDENTIFIER ::= { id-sti 4 }

 END -- of DOMSECSyntax
```

8. References

[1] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[4] International Telecommunications Union, Recommendation X.208, "Open systems interconnection: specification of Abstract Syntax Notation (ASN.1)", CCITT Blue Book, 1989.

[5] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

9. Authors' Addresses

Tim Dean
QinetiQ
St. Andrews Road
Malvern
Worcs
WR14 3PS

Phone: +44 (0) 1684 894239
Fax: +44 (0) 1684 896660
EMail: tbdean@QinetiQ.com

William Ottaway
QinetiQ
St. Andrews Road
Malvern
Worcs
WR14 3PS

Phone: +44 (0) 1684 894079
Fax: +44 (0) 1684 896660
EMail: wjottaway@QinetiQ.com

10. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.