Domain-based signing and encryption using S/MIME
draft-melnikov-smime-msa-to-mda-04

Abstract

   The S/MIME protocols Message Specification (RFC 5751), Cryptographic
   Message Syntax (RFC 5652), S/MIME Certificate Handling (RFC 5750) and
   Enhanced Security Services for S/MIME (RFC 2634) specify a consistent
   way to securely send and receive MIME messages providing end to end
   integrity, authentication, non-repudiation and confidentiality. This
   document identifies a number of interoperability, technical,
   procedural and policy related issues that may result in end-to-end
   security services not being achievable. To resolve such issues, this
   document profiles domain-based signing and encryption using S/MIME,
   such as specifying how S/MIME signing and encryption can be applied
   between a Message Submission Agent (MSA) and a Message Delivery Agent
   (MDA) or between 2 Message Transfer Agents (MTA).

   This document is also registering 2 URI scheme: "smtp" and "submit"
   which are used for designating SMTP/SMTP Submission servers
   (respectively), as well as SMTP/Submission client accounts.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on September 6, 2014.

Copyright Notice

Table of Contents

1. Introduction

   The S/MIME [RFC5750][RFC5751] 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 [RFC5322], or when a single
   domain wishes to communicate securely with one of its members
   residing on an untrusted domain. The main scenario covered by this
   document is domain-to-domain, although it is also applicable to
   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 (MUA-to-MUA) 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. Message screening and audit: Server-based mechanisms such as
       searching for prohibited words or other unauthorized content,
       virus scanning, and audit, are incompatible with end-to-end
       encryption. It is generally not acceptable to allow content in/
       out of an organization without checking, so boundary decryption
       is vital.

2. 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.

3. Heterogeneous message formats: One organization using X.400
    series protocols wishes to communicate with another using SMTP
    [RFC5321]. Message reformatting at gateways makes end-to-end
    encryption and signature validation impossible.

4. 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.

5. Problems deploying fully S/MIME capable email clients on some
    platforms. Signature verification at a border MTA can be coupled
    with use of Authentication-Results header field [RFC7001] to
    convey results of verification.

This document describes an approach to solving these problems by
providing message security services at the level of a domain or an
organization. Such domain-based or organization-based message
security services are referred to as domain security services. 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/mobile device. For
example, a domain may decide to provide MUA-to-MUA signatures but
domain-to-domain encryption services. Or it may allow MUA-to-MUA
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 is a

specification for how domain-based S/MIME signing and encryption can
be applied in different ways to meet different needs in different
circumstances.

Message Transfer Agents (MTAs), Message Submission Agents (MSAs),
Message Delivery Agents (MDAs), guards, firewalls and protocol
translation gateways can provide domain security services. As with
MUA 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.

2. Conventions Used in This Document

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 [RFC2119].

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').

Domain-based Sending Agent - an MSA or sending domain MTA performing
Domain-based service(s).

Domain-based Receiving Agent - a receiving domain MTA or MDA
performing Domain-based service(s).

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 document as a "Domain Confidentiality Authority"

(DCA). As per above, a DCA can be a Domain-based Sending Agent or a
Domain-based Receiving Agent.

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 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 [RFC2634].

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.

2.5. Signature Encapsulation

ESS [RFC2634] 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 can 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 [RFC5652]. However, the eContent field
    will contain the unsigned message instead of being left empty as
    suggested in section 5.2 in CMS [RFC5652]. 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.

2.6. Naming Conventions

The subject name of the Domain-based sending agent's X.509
certificate is not restricted as specified in RFC 3183 [RFC3183]. In
order for a verifier to recognize a signing/encrypting certificate as
the Domain-based sending agent's certificate, it MUST contain
uniformResourceIdentifier GeneralName of the format "<protocol
>://<fully-qualified-domain>" and/or dNSName of the format <fully-
qualified-domain> in its SubjectAltName [RFC5280]. (Here <fully-
qualified-domain> is the domain that is being served by the signing/
encrypting MSA/MTA. <protocol> is "submit" for MSAs and "smtp" for
MTAs.)

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 to the user.

This naming rule prevents agents from one organization masquerading
as domain signing or encryption authorities on behalf of another.
For the other types of signature defined in future documents, no such
namin rule is defined.

Implementations conforming to this standard MUST support this naming
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 to the user.

Note that a X.509 certificate of a signing Domain-based sending agent
can be distinguished from a certificate of encrypting domain-based
sending agent by checking for keyUsage as specified in [RFC5280]
   Section 4.2.1.3.

3. Domain-Based S/MIME Signing

3.1. General

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
Section 3.2. On receiving a message containing this attribute, the
naming convention (see Section 2.6) checks are invoked.

Implementations conforming to this standard MUST support the naming
convention specified in Section 2.6 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.

3.2. 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 Section 2.6. 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 following object identifier identifies the SignatureType
attribute:

id-aa-signatureType OBJECT IDENTIFIER ::= { iso(1)
          member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) 28 }

The ASN.1 [ASN.1] 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 [RFC5652]. 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 [RFC5652]. 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.

id-sti-delegatedOriginatorSig OBJECT IDENTIFIER ::= { id-sti 5 }
          -- delegated originator 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 and the delegated
originator types, MUST encapsulate other signatures. Note a DOMSEC
defined signature could be encapsulating an empty signature as
defined in Section 2.5.

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 [RFC5652], 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.

3.3. **Domain Signature Generation and Verification**

   A 'domain signature' is a signature generated on behalf of a set of
   users who belong to the specific domain. The signature MUST adhere
   to the naming conventions in Section 2.6. A 'domain signature' on a
   message authenticates the fact that the message has been released
   from that domain. (It also provides integrity and non-repudiation of
   message between Domain-based Sending Agents and Receiving Agents.)
   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 SMTP
   authentication credentials [RFC4954], authentication provided by
   STARTTLS [RFC3207] (possibly combined with SMTP authentication), the
   physical address of the originating client or an authenticated IP
   link, etc.

   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 Section 2.6.

   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 the
   naming convention has been used as specified in Section 2.6.

   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.

3.4. 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 that follows the naming convention specified in
Section 2.6. 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 [RFC2634] 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 can be multiple 'additional attributes' signatures in an S/MIME
encoding.

3.5. 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, an agent
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 review signature MUST do so using a
certificate that follows the naming convention specified in
Section 2.6. 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 can be multiple review signatures in an S/MIME encoding.

3.6. 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.

3.7. Delegated Originator Signature

The 'delegated originator signature' is similar to the 'domain
signature' (Section 3.3), but it also indicates that MSA signed
message with a unique originator-specific key.

If the originator's authenticity is successfully verified as
specified in Section 3.3 and all other signatures present are valid,
including those that have been encrypted, a 'delegated originator
signature' can be added to a message.

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

An entity generating a delegated originator signature MUST do so
using a certificate that follows the naming convention specified in
Section 2.6. On reception, a check MUST be made to ensure that the
naming convention has been used.

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

A delegated originator signature is indicated by the presence of the
value id-sti-delegatedOriginatorSig in a 'signature type' signed
attribute.

4. Domain-based S/MIME Encryption and Decryption

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 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 [RFC5652]. 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
[RFC5652]. 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 is 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 [RFC5652]. 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. 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 [RFC5652].
A DCA encryption agent MUST be named according to the naming
convention specified in Section 2.6. 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 [RFC4510] 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 Section 2.6. The user certificate
    attribute associated with the DCA's directory entry is then
    obtained.

This document does not mandate either of these processes. Whichever
one is used, the naming 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 [RFC5652].

DCA encryption may be performed for decryption by the end recipient
and/or by a DCA. End recipient decryption is described in CMS
[RFC5652]. DCA decryption is described in Section 4.2.

4.2. 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 [RFC5652]; in the second case, the encryption
process is described in Section 4.1.

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 [RFC2634] 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), 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:

o The "outer" signedData layer that includes an mlExpansionHistory
   attribute or encapsulates an envelopedData object.

o An envelopedData layer.

o 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.

6. 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. Strip off all the signedData layers down to the envelopedData
    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. IANA Considerations

This document registers 2 URI schemes, described in subsections of
this section. IANA is requested to add them to the list of Permanent
URI schemes.

6.1. SMTP URI registration

URI scheme name: smtp

Status: permanent

URI scheme syntax:
smtpuri = "smtp://" authority ["/" ["?" query]]
authority = <defined in RFC 3986>
query = <defined in RFC 3986>
If :<port> is omitted from authority, the port defaults to 25.
The query component is reserved for future extensions.

URI scheme semantics:
The smtp: URI scheme is used to designate SMTP servers (e.g.
listener endpoints, S/MIME agents performing Domain signing), SMTP
accounts.
There is no MIME type associated with this URI.

Encoding considerations:

SMTP user names are UTF-8 strings and MUST be percent-encoded as
required by the URI specification [RFC3986], Section 2.1.

Applications/protocols that use this URI scheme name:
The smtp: URI is intended to be used by applications that might
need access to an SMTP server (for example email clients or MTAs)
or for SMTP servers to describe their listener endpoints.

Interoperability considerations:
Several implementations are already using smtp: URIs for server
configuration.

Security considerations: Clients resolving smtp: URIs that wish to
achieve data confidentiality and/or integrity SHOULD use the
STARTTLS command (if supported by the server) before starting
authentication, or use a SASL mechanism, such as GSSAPI, that
provides a confidentiality security layer.

Contact: Alexey Melnikov <alexey.melnikov@isode.com>

Author/Change controller: IESG

References: [[This document]] and [RFC5321].

6.2. SUBMIT URI registration

URI scheme name: submit

Status: permanent

URI scheme syntax:
submituri = "submit://" authority ["/" ["?" query]]
authority = <defined in RFC 3986>
query = <defined in RFC 3986>
If :<port> is omitted from authority, the port defaults to 587.
The query component is reserved for future extensions.

URI scheme semantics:
The submit: URI scheme is used to designate SMTP Submission
servers (e.g. listener endpoints, S/MIME agents performing Domain
signing), SMTP accounts.
There is no MIME type associated with this URI.

Encoding considerations:
SMTP user names are UTF-8 strings and MUST be percent-encoded as
required by the URI specification [RFC3986], Section 2.1.

        Applications/protocols that use this URI scheme name:
        The submit: URI is intended to be used by applications that might
        need access to an SMTP Submission server (for example email
        clients) or for SMTP Submission servers to describe their listener
        endpoints.

        Interoperability considerations:
        None.

        Security considerations: Clients resolving submit: URIs that wish
        to achieve data confidentiality and/or integrity SHOULD use the
        STARTTLS command (if supported by the server) before starting
        authentication, or use a SASL mechanism, such as GSSAPI, that
        provides a confidentiality security layer.

        Contact: Alexey Melnikov <alexey.melnikov@isode.com>

        Author/Change controller: IESG

        References: [[This document]] and [RFC6409].

7. Security Considerations

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

   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.

8. References

8.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2634]   Hoffman, P., "Enhanced Security Services for S/MIME", RFC
            2634, June 1999.

[RFC5652]   Housley, R., "Cryptographic Message Syntax (CMS)", STD 70,
            RFC 5652, September 2009.

[RFC5280]   Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
            Housley, R., and W. Polk, "Internet X.509 Public Key
            Infrastructure Certificate and Certificate Revocation List
            (CRL) Profile", RFC 5280, May 2008.

[RFC5750]   Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet
            Mail Extensions (S/MIME) Version 3.2 Certificate
            Handling", RFC 5750, January 2010.

[RFC5751]   Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet
            Mail Extensions (S/MIME) Version 3.2 Message
            Specification", RFC 5751, January 2010.

[RFC5321]   Klensin, J., "Simple Mail Transfer Protocol", RFC 5321,
            October 2008.

[RFC6409]   Gellens, R. and J. Klensin, "Message Submission for Mail",
            STD 72, RFC 6409, November 2011.

[RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
            Resource Identifier (URI): Generic Syntax", STD 66, RFC
            3986, January 2005.

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

8.2. Informative References

[RFC3207]   Hoffman, P., "SMTP Service Extension for Secure SMTP over
            Transport Layer Security", RFC 3207, February 2002.

   [RFC4954] Siemborski, R. and A. Melnikov, "SMTP Service Extension
              for Authentication", RFC 4954, July 2007.

   [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322,
              October 2008.

   [RFC3183] Dean, T. and W. Ottaway, "Domain Security Services using S
              /MIME", RFC 3183, October 2001.

   [RFC4510] Zeilenga, K., "Lightweight Directory Access Protocol
              (LDAP): Technical Specification Road Map", RFC 4510, June
              2006.

   [RFC7001] Kucherawy, M., "Message Header Field for Indicating
              Message Authentication Status", RFC 7001, September 2013.

Appendix A. Changes from RFC 3183

   Unlike RFC 3183, subject names of domain signing/encrypting X.509
   certificates don't have to have a specific form. But Subject
   Alternative Names need to include URIs for domain being protected.

   A new signature type was added for the case when MSA signs/encrypts a
   message on behalf of a user with a user specific key.

   Incorporated erratum 3757 resolution.

   Updated references and some minor editorial corrections.

Appendix B. Acknowledgements

   This document contains lots of text from RFC 3183.

   Editors would like to thank Steve Kille, David Wilson, Alan Ross and
   Vijay K. Gurbani for comments and corrections.

Authors' Addresses

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

   EMail: wjottaway@QinetiQ.com

   Alexey Melnikov (editor)
   Isode Ltd
   5 Castle Business Village
   36 Station Road
   Hampton, Middlesex TW12 2BX
   UK

   EMail: Alexey.Melnikov@isode.com