SIP-based Messaging with S/MIME
draft-campbell-sip-messaging-smime-05

Abstract

Mobile messaging applications used with the Session Initiation Protocol (SIP) commonly use some combination of the SIP MESSAGE method and the Message Session Relay Protocol (MSRP). While these provide mechanisms for hop-by-hop security, neither natively provides end-to-end protection. This document offers guidance on how to provide end-to-end authentication, integrity protection, and confidentiality using the Secure/Multipurpose Internet Mail Extensions (S/MIME). It updates and provides clarifications for RFC 3261, RFC 3428, and RFC 4975.

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 https://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 July 30, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................... 4
3. Problem Statement and Scope ............................. 4
4. Applicability of S/MIME ..................................... 5
  4.1. Signed Messages ........................................... 5
  4.2. Encrypted Messages ...................................... 6
  4.3. Signed and Encrypted Messages ......................... 7
  4.4. Certificate Handling .................................... 8
       4.4.1. Subject Alternative Name ......................... 8
       4.4.2. Certificate Validation .......................... 8
5. Transfer Encoding ............................................ 8
6. User Agent Capabilities ...................................... 9
7. Using S/MIME with the SIP MESSAGE Method ............. 10
  7.1. Size Limit ................................................ 10
  7.2. SIP User Agent Capabilities .......................... 10
  7.3. Failure Cases ............................................ 11
8. Using S/MIME with MSRP ..................................... 11
  8.1. Chunking .................................................. 11
  8.2. Streamed Data ............................................ 12
  8.3. Indicating support for S/MIME ........................ 12
  8.4. MSRP URIs ............................................... 13
  8.5. Failure Cases ............................................ 13
9. S/MIME Interaction with other SIP Messaging Features ... 14
  9.1. Common Profile for Instant Messaging ............... 14
  9.2. Instant Message Delivery Notifications ............ 15
10. Examples ...................................................... 15
  10.1. Signed Message in SIP Including the Sender’s Certificate 16
  10.2. Signed Message in SIP with No Certificate .......... 17
  10.3. MSRP Signed and Encrypted Message in a Single Chunk .. 18
  10.4. MSRP Signed and Encrypted Message sent in Multiple
        Chunks ................................................. 20
11. IANA Considerations ......................................... 22
12. Security Considerations ..................................... 22
13. References .................................................. 24
  13.1. Normative References ................................... 24
  13.2. Informative References ............................... 26
Appendix A. Message Details .................................. 28
  A.1. Signed Message .......................................... 28
  A.2. Short Signed Message ................................... 30
  A.3. Signed and Encrypted Message ........................ 31
1. Introduction

Several Mobile Messaging systems use the Session Initiation Protocol (SIP) \[RFC3261\], typically as some combination of the SIP MESSAGE method \[RFC3428\] and the Message Session Relay Protocol (MSRP) \[RFC4975\]. For example, Voice over LTE (VoLTE) uses the SIP MESSAGE method to send Short Message Service (SMS) messages. The Open Mobile Alliance (OMA) Converged IP Messaging (CPM) \[CPM\] system uses the SIP MESSAGE Method for short "pager mode" messages and MSRP for large messages and for sessions of messages. The GSM Association (GMSA) rich communication services (RCS) uses CPM for messaging \[RCS\].

At the same time, organizations increasingly depend on mobile messaging systems to send notifications to their customers. Many of these notifications are security-sensitive. For example, such notifications are commonly used for notice of financial transactions, notice of login or password change attempts, and sending of two-factor authentication codes.

Both SIP and MSRP can be used to transport any content using Multipurpose Internet Mail Extensions (MIME) formats. The SIP MESSAGE method is typically limited to short messages (under 1300 octets for the MESSAGE request). MSRP can carry arbitrarily large messages, and can break large messages into chunks.

While both SIP and MSRP provide mechanisms for hop-by-hop security, neither provides native end-to-end protection. Instead, they depend on S/MIME \[I-D.ietf-lamps-rfc5750-bis\],[I-D.ietf-lamps-rfc5751-bis\]. However at the time of this writing, S/MIME is not in common use for SIP- and MSRP-based messaging services. This document updates and clarifies RFC 3261, RFC 3428, and RFC 4975 in an attempt to make S/MIME for SIP and MSRP easier to implement and deploy in an interoperable fashion.

This document updates RFC 3261, RFC 3428, and RFC 4975 to update the cryptographic algorithm recommendations and the handling of S/MIME data objects. It updates RFC 3261 to allow S/MIME signed messages to be sent without imbedded certificates in some situations. Finally, it updates RFC 3261, RFC 3428, and RFC 4975 to clarify error reporting requirements for certain situations.
2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Problem Statement and Scope

This document discusses the use of S/MIME with SIP-based messaging. Other standardized messaging protocols exist, such as the Extensible Messaging and Presence Protocol (XMPP) [RFC6121]. Likewise, other end-to-end protection formats exist, such as JSON Web Signatures [RFC7515] and JSON Web Encryption [RFC7516].

This document focuses on SIP-based messaging because its use is becoming more common in mobile environments. It focuses on S/MIME since several mobile operating systems already have S/MIME libraries installed. While there may also be value in specifying end-to-end security for other messaging and security mechanisms, it is out of scope for this document.

MSRP sessions are negotiated using the Session Description Protocol (SDP) [RFC4566] offer/answer mechanism [RFC3264] or similar mechanisms. This document assumes that SIP is used for the offer/answer exchange. However, the techniques should be adaptable to other signaling protocols.

[RFC3261], [RFC3428], and [RFC4975] already describe the use of S/MIME. [RFC3853] updates SIP to support the Advanced Encryption Standard (AES). In aggregate that guidance is incomplete, contains inconsistencies, and is still out of date in terms of supported and recommended algorithms.

The guidance in RFC 3261 is based on an implicit assumption that S/MIME is being used to secure signaling applications. That advice is not entirely appropriate for messaging application. For example, it assumes that message decryption always happens before the SIP transaction completes.

This document offers normative updates and clarifications to the use of S/MIME with the SIP MESSAGE method and MSRP. It does not attempt to define a complete secure messaging system. Such system would require considerable work around user enrollment, certificate and key generation and management, multiparty chats, device management, etc. While nothing herein should preclude those efforts, they are out of scope for this document.
This document primarily covers the sending of single messages, for example "pager-mode messages" sent using the SIP MESSAGE method and "large messages" sent in MSRP. Techniques to use a common signing or encryption key across a session of messages are out of scope for this document.

Cryptographic algorithm requirements in this document are intended to supplement those already specified for SIP and MSRP.

4. Applicability of S/MIME

The Cryptographic Message Syntax (CMS) [RFC5652] is an encapsulation syntax that is used to digitally sign, digest, authenticate, or encrypt arbitrary message content. The CMS supports a variety of architectures for certificate-based key management, especially the one defined by the IETF PKIX (Public Key Infrastructure using X.509) working group [RFC5280]. The CMS values are generated using ASN.1 [X680], using the Basic Encoding Rules (BER) and Distinguished Encoding Rules (DER) [X690].

The S/MIME Message Specification [I-D.ietf-lamps-rfc5751-bis] defines MIME body parts based on the CMS. In this document, the application/pkcs7-mime media type is used to digitally sign an encapsulated body part, and it is also used to encrypt an encapsulated body part.

4.1. Signed Messages

While both SIP and MSRP require support for the multipart/signed format, the use of application/pkcs7-mime is RECOMMENDED for most signed messages. Experience with the use of S/MIME in electronic mail has shown that multipart/signed bodies are at greater risk of "helpful" tampering by intermediaries, a common cause of signature validation failure. This risk is also present for messaging applications; for example, intermediaries might insert Instant Message Delivery notification requests into messages (see Section 9.2). The application/pkcs7-mime format is also more compact, which can be important for messaging applications, especially when using the SIP MESSAGE method (see Section 7.1). The use of multipart/signed may still make sense if the message needs to be readable by receiving agents that do not support S/MIME.

When generating a signed message, sending user agents (UA) SHOULD follow the conventions specified in [I-D.ietf-lamps-rfc5751-bis] for the application/pkcs7-mime media type with smime-type=signed-data. When validating a signed message, receiving UAs MUST follow the conventions specified in [I-D.ietf-lamps-rfc5751-bis] for the application/pkcs7-mime media type with smime-type=signed-data.
Sending and receiving UAs MUST support the SHA-256 message digest algorithm [RFC5754]. For convenience, the SHA-256 algorithm identifier is repeated here:

\[
\text{id-sha256 OBJECT IDENTIFIER ::= {joint-iso-itu-t(2)country(16)us(840)organization(1)gov(101)csor(3)nistalgorithm(4)hashalgs(2)1}}
\]

Sending and receiving UAs MAY support other message digest algorithms.

Sending and receiving UAs MUST support the Elliptic Curve Digital Signature Algorithm (ECDSA) using the NIST P-256 elliptic curve and the SHA-256 message digest algorithm [RFC5480][RFC5753]. Sending and receiving UAs SHOULD support the Edwards-curve Digital Signature Algorithm (EdDSA) with curve25519 (Ed25519) [RFC8032][RFC8419]. For convenience, the ECDSA with SHA-256 algorithm identifier, the object identifier for the well-known NIST P-256 elliptic curve, and the Ed25519 algorithm identifier are repeated here:

\[
\text{ecdsa-with-SHA256 OBJECT IDENTIFIER ::= {iso(1)member-body(2)us(840)ansi-X9-62(10045)signatures(4)ecdsa-with-SHA2(3)2}}
\]

-- Note: the NIST P-256 elliptic curve is also known as secp256r1.

\[
\text{secp256r1 OBJECT IDENTIFIER ::= {iso(1)member-body(2)us(840)ansi-X9-62(10045)curves(3)prime(1)7}}
\]

\[
\text{id-Ed25519 OBJECT IDENTIFIER ::= {iso(1)identified-organization(3)thawte(101)112}}
\]

### 4.2. Encrypted Messages

When generating an encrypted message, sending UAs MUST follow the conventions specified in [I-D.ietf-lamps-rfc5751-bis] for the application/pkcs7-mime media type with smime-type=auth-enveloped-data. When decrypting a received message, receiving UAs MUST follow the conventions specified in [I-D.ietf-lamps-rfc5751-bis] for the application/pkcs7-mime media type with smime-type=auth-enveloped-data.

Sending and receiving UAs MUST support the AES-128-GCM for content encryption [RFC5084]. For convenience, the AES-128-GCM algorithm identifier is repeated here:
id-aes128-GCM OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
cisor(3) nistAlgorithm(4) aes(1) 6 }

Sending and receiving UAs MAY support other content authenticated encryption algorithms.

Sending and receiving UAs MUST support the AES-128-WRAP algorithm for encryption of one AES key with another AES key [RFC3565]. For convenience, the AES-128-WRAP algorithm identifier is repeated here:

id-aes128-wrap OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
cisor(3) nistAlgorithm(4) aes(1) 5 }

Sending and receiving UAs MAY support other key encryption algorithms.

Symmetric key-encryption keys can be distributed before messages are sent. If sending and receiving UAs support previously distributed key-encryption keys, then they MUST assign a KEKIdentifier [RFC5652] to the previously distributed symmetric key.

Alternatively, a key agreement algorithm can be used to establish a single-use key-encryption key. If sending and receiving UAs support key agreement, then they MUST support the Elliptic Curve Diffie-Hellman (ECDH) algorithm using the NIST P-256 elliptic curve and the ANSI-X9.63-KDF key derivation function with the SHA-256 message digest algorithm [RFC5753]. If sending and receiving UAs support key agreement, then they SHOULD support the Elliptic Curve Diffie-Hellman (ECDH) algorithm using curve25519 (X25519) [RFC7748][RFC8418]. For convenience, the identifiers for the ECDH algorithm using the ANSI-X9.63-KDF with SHA-256 algorithm and for the X25519 algorithm are repeated here:

dhSinglePass-stdDH-sha256kdf-scheme OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) certicom(132)
schemes(1) 11 1 }

id-X25519 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) thawte(101) 110 }

4.3. Signed and Encrypted Messages

RFC 3261 section 23.2 says that when a User Agent Client (UAC) sends signed and encrypted data, it "SHOULD" send an EnvelopedData object encapsulated within a SignedData message. That essentially says that one should encrypt first, then sign. This document updates RFC 3261
to say that, when sending signed and encrypted user content in a SIP 
MESSAGE request, the sending UAs MUST sign the message first, and then encrypt it. That is, it must send the SignedData object inside an AuthEnvelopedData object. For interoperability reasons, recipients SHOULD accept messages signed and encrypted in either order.

4.4. Certificate Handling

Sending and receiving UAs MUST follow the S/MIME certificate handling procedures [I-D.ietf-lamps-rfc5750-bis], with a few exceptions detailed below.

4.4.1. Subject Alternative Name

In both SIP and MSRP, the identity of the sender of a message is typically expressed as a SIP URI.

The subject alternative name extension is used as the preferred means to convey the SIP URI of the subject of a certificate. Any SIP URI present MUST be encoded using the uniformResourceIdentifier CHOICE of the GeneralName type as described in [RFC5280], Section 4.2.1.6. Since the SubjectAltName type is a SEQUENCE OF GeneralName, multiple URIs MAY be present.

Other methods of identifying a certificate subject MAY be used.

4.4.2. Certificate Validation

When validating a certificate, receiving UAs MUST support the Elliptic Curve Digital Signature Algorithm (ECDSA) using the NIST P-256 elliptic curve and the SHA-256 message digest algorithm [RFC5480].

Sending and receiving UAs MAY support other digital signature algorithms for certificate validation.

5. Transfer Encoding

SIP and MSRP UAs are always capable of receiving binary data. Inner S/MIME entities do not require base64 encoding [RFC4648].

Both SIP and MSRP provide 8-bit safe transport channels; base64 encoding is not generally needed for the outer S/MIME entities. However, if there is a chance a message might cross a 7-bit transport (for example, gateways that convert to a 7-bit transport for intermediate transfer), base64 encoding may be needed for the outer entity.
6. User Agent Capabilities

Messaging UAs may implement a subset of S/MIME capabilities. Even when implemented, some features may not be available due to configuration. For example, UAs that do not have user certificates cannot sign messages on behalf of the user or decrypt encrypted messages sent to the user. At a minimum, a UA that supports S/MIME MUST be able to validate a signed message.

End-user certificates have long been a barrier to large-scale S/MIME deployment. But since UAs can validate signatures even without local certificates, the use case of organizations sending secure notifications to their users becomes a sort of "low hanging fruit". That being said, the signed-notification use case still requires shared trust anchors.

SIP and MSRP UAs advertise their level of support for S/MIME by indicating their capability to receive the "application/pkcs7-mime" media type.

The fact that a UA indicates support for the "multipart/signed" media type does not necessarily imply support for S/MIME. The UA might just be able to display clear-signed content without validating the signature. UAs that wish to indicate the ability to validate signatures for clear-signed messages MUST also indicate support for "application/pkcs7-signature".

A UA can indicate that it can receive all smime-types by advertising "application/pkcs7-mime" with no parameters. If a UA does not accept all smime-types, it advertises the media type with the appropriate parameters. If more than one are supported, the UA includes a separate instance of the media-type string, appropriately parameterized, for each.

For example, a UA that can only receive signed-data would advertise "application/pkcs7-mime; smime-type=signed-data".

SIP signaling can fork to multiple destinations for a given Address of Record (AoR). A user might have multiple UAs with different capabilities; the capabilities remembered from an interaction with one such UA might not apply to another. (See Section 7.2.)

UAs can also advertise or discover S/MIME using out-of-band mechanisms. Such mechanisms are beyond the scope of this document.
7. Using S/MIME with the SIP MESSAGE Method

The use of S/MIME with the SIP MESSAGE method is described in section 11.3 of [RFC3428], and for SIP in general in section 23 of [RFC3261]. This section and its child sections offer clarifications for the use of S/MIME with the SIP MESSAGE method, along with related updates to RFC 3261 and RFC 3428.

7.1. Size Limit

SIP MESSAGE requests are typically limited to 1300 octets. That limit applies to the entire message, including both SIP header fields and the message content. This is due to the potential for fragmentation of larger requests sent over UDP. In general, it is hard to be sure that no proxy or other intermediary will forward a SIP request over UDP somewhere along the path. Therefore, S/MIME messages sent via SIP MESSAGE should be kept as small as possible. Messages that will not fit within the limit can be sent using MSRP.

Section 23.2 of [RFC3261] says that a SignedData message must contain a certificate to be used to validate the signature. In order to reduce the message size, this document updates that to say that a SignedData message sent in a SIP MESSAGE request SHOULD contain the certificate, but MAY omit it if the sender has reason to believe that the recipient already has the certificate in its keychain, or has some other method of accessing the certificate.

7.2. SIP User Agent Capabilities

SIP user agents (UA) can theoretically indicate support for S/MIME by including the appropriate media type or types in the SIP Accept header field in a response to an OPTIONS request, or in a 415 response to a SIP request that contained an unsupported media type in the body. Unfortunately, this approach may not be reliable in the general case. In the case where a downstream SIP proxy forks an OPTIONS or other non-INVITE request to multiple UASs, that proxy will only forward the "best" response. If the recipient has multiple devices, the sender may only learn the capabilities of the device that sent the forwarded response. Blindly trusting this information could result in S/MIME messages being sent to UAs that do not support it, which would be at best confusing and at worst misleading to the recipient.

UAs might be able to use the user agent capabilities framework [RFC3840] to indicate support. However doing so would require the registration of one or more media feature tags with IANA.
UAs MAY use other out-of-band methods to indicate their level of support for S/MIME.

7.3. Failure Cases

Section 23.2 of [RFC3261] requires that the recipient of a SIP request that includes a body part of an unsupported media type and a Content-Disposition header field "handling" parameter of "required" return a 415 "Unsupported Media Type" response. Given that SIP MESSAGE exists for no reason other than to deliver content in the body, it is reasonable to treat the top-level body part as always required. However [RFC3428] makes no such assertion. This document updates section 11.3 [RFC3428] to add the statement that a UAC that receives a SIP MESSAGE request with an unsupported media type MUST return a 415 "Unsupported Media Type" response.

Section 23.2 of [RFC3261] says that if a recipient receives an S/MIME body encrypted to the wrong certificate, it MUST return a SIP 493 (Undecipherable) response, and SHOULD send a valid certificate in that response. This is not always possible in practice for SIP MESSAGE requests. The User Agent Server (UAS) may choose not to decrypt a message until the user is ready to read it. Messages may be delivered to a message store, or sent via a store-and-forward service. This document updates RFC 3261 to say that the UAS SHOULD return a SIP 493 response if it immediately attempts to decrypt the message and determines the message was encrypted to the wrong certificate. However, it MAY return a 200-class response if decryption is deferred.

8. Using S/MIME with MSRP

MSRP has features that interact with the use of S/MIME. In particular, the ability to send messages in chunks, the ability to send messages of unknown size, and the use of SDP to indicate media-type support create considerations for the use of S/MIME.

8.1. Chunking

MSRP allows a message to be broken into "chunks" for transmission. In this context, the term "message" refers to an entire message that one user might send to another. A chunk is a fragment of that message sent in a single MSRP SEND request. All of the chunks that make up a particular message share the same Message-ID value.

The sending user agent may break a message into chunks, which the receiving user agent will reassemble to form the complete message. Intermediaries such as MSRP Relays [RFC4976] might break chunks into smaller chunks, or might reassemble chunks into larger ones;
therefore the message received by the recipient may be broken into a
different number of chunks than were sent by the recipient.
Intermediaries might also cause chunks to be received in a different
order than sent.

The sender MUST apply any S/MIME operations to the whole message
prior to breaking it into chunks. Likewise, the receiver needs to
reassemble the message from its chunks prior to decrypting,
validating a signature, etc.

MSRP chunks are framed using an end-line. The end-line comprises
seven hyphens, a 64-bit random value taken from the start line, and a
continuation flag. MSRP requires the sending user agent to scan data
sent in a specific chunk to be sent ensure that the end-line does not
accidentally occur as part of the sent data. This scanning occurs on
a chunk rather than a whole message, consequently it must occur after
the sender applies any S/MIME operations.

8.2. Streamed Data

MSRP allows a mode of operation where a UA sends some chunks of a
message prior to knowing the full length of the message. For
example, a sender might send streamed data over MSRP as a single
message, even though it doesn't know the full length of that data in
advance. This mode is incompatible with S/MIME, since a sending UA
must apply S/MIME operations to the entire message in advance of
breaking it into chunks.

Therefore, when sending a message in an S/MIME format, the sender
MUST include the Byte-Range header field for every chunk, including
the first chunk. The Byte-Range header field MUST include the total
length of the message.

A higher layer could choose to break such streamed data into a series
of messages prior to applying S/MIME operation, so that each fragment
appears as a distinct S/MIME separate message in MSRP. Such
mechanisms are beyond the scope for this document.

8.3. Indicating support for S/MIME

A UA that supports this specification MUST explicitly include the
appropriate media type or types in the "accept-types" attribute in
any SDP offer or answer that proposes MSRP. It MAY indicate that it
requires S/MIME wrappers for all messages by putting appropriate
S/MIME media types in the "accept-types" attribute and putting all
other supported media types in the "accept-wrapped-types" attribute.
For backwards compatibility, a sender MAY treat a peer that includes an asterisk ("*") in the "accept-types" attribute as potentially supporting S/MIME. If the peer returns an MSRP 415 response to an attempt to send an S/MIME message, the sender should treat the peer as not supporting S/MIME for the duration of the session, as indicated in [RFC4975].

While these SDP attributes allow an endpoint to express support for certain media types only when wrapped in a specified envelope type, it does not allow the expression of more complex structures. For example, an endpoint can say that it supports text/plain and text/html, but only when inside an application/pkcs7 or message/cpim container, but it cannot express a requirement for the leaf types to always be contained in an application/pkcs7 container nested inside a message/cpim container. This has implications for the use of S/MIME with the message/cpim format. (See Section 9.1.)

MSRP allows multiple reporting modes that provide different levels of feedback. If the sender includes a Failure-Report header field with a value of "no", it will not receive failure reports. This mode should not be used carelessly, since such a sender would never see a 415 response as described above, and would have no way to learn that the recipient could not process an S/MIME body.

8.4. MSRP URIs

MSRP URIs are ephemeral. Endpoints MUST NOT use MSRP URIs to identify certificates, or insert MSRP URIs into certificate Subject Alternative Name fields. When MSRP sessions are negotiated using SIP [RFC3261], the SIP AoRs of the peers are used instead.

Note that MSRP allows messages to be sent between peers in either direction. A given MSRP message might be sent from the SIP offerer to the SIP answerer. Thus, the sender and recipient roles may reverse between one message and another in a given session.

8.5. Failure Cases

Successful delivery of an S/MIME message does not indicate that the recipient successfully decrypted the contents or validated a signature. Decryption and/or validation may not occur immediately on receipt, since the recipient may not immediately view the message, and the user agent may choose not to attempt decryption or validation until the user requests it.

Likewise, successful delivery of S/MIME enveloped data does not, on its own, indicate that the recipient supports the enclosed media type. If the peer only implicitly indicated support for the enclosed
media type through the use of a wildcard in the "accept-types" or "accept-wrapped types" SDP attributes, it may not decrypt the message in time to send a 415 response.

9. S/MIME Interaction with other SIP Messaging Features

9.1. Common Profile for Instant Messaging

The Common Profile for Instant Messaging (CPIM) [RFC3860] defines an abstract messaging service, with the goal of creating gateways between different messaging protocols that could relay instant messages without change. The SIP MESSAGE method and MSRP were initially designed to map to the CPIM abstractions. However, at the time of this writing, CPIM-compliant gateways have not been deployed. To the authors’ knowledge, no other IM protocols have been explicitly mapped to CPIM.

CPIM also defines the abstract messaging URI scheme "im:". As of the time of this writing, the "im:" scheme is not in common use.

The Common Profile for Instant Messages Message Format [RFC3862] allows UAs to attach transport-neutral metadata to arbitrary MIME content. The format was designed as a canonicalization format to allow signed data to cross protocol-converting gateways without loss of metadata needed to verify the signature. While it has not typically been used for that purpose, it has been used for other metadata applications, for example, Instant Message Delivery Notifications (IMDN) [RFC5438] and MSRP Multi-party Chat [RFC7701].

In the general case, a sender applies end-to-end signature and encryption operations to the entire MIME body. However, some messaging systems expect to inspect and in some cases add or modify metadata in CPIM header fields. For example, CPM- and RCS-based services include application servers that may need to insert time stamps into chat messages, and may use additional metadata to characterize the content and purpose of a message to determine application behavior. The former will cause validation failure for signatures that cover CPIM metadata, while the latter is not possible if the metadata is encrypted. Clients intended for use in such networks MAY choose to apply end-to-end signatures and encryption operations to only the CPIM payload, leaving the CPIM metadata unprotected from inspection and modification. UAs that support S/MIME and CPIM SHOULD be able validate signatures and decrypt enveloped data both when those operations are applied to the entire CPIM body, and when they are applied to just the CPIM payload. This means that the receiver needs to be flexible in its MIME document parsing and that it cannot make assumptions that S/MIME protected
body parts will always be in the same position or level in the message payload.

If such clients need to encrypt or sign CPIM metadata end-to-end, they can nest a protected CPIM message format payload inside an unprotected CPIM message envelope.

The use of CPIM metadata fields to identify certificates or to authenticate SIP or MSRP header fields is out of scope for this document.

9.2. Instant Message Delivery Notifications

The Instant Message Delivery Notification (IMDN) mechanism [RFC5438] allows both endpoints and intermediary application servers to request and to generate delivery notifications. The use of S/MIME does not impact strictly end-to-end use of IMDN. IMDN recommends that devices that are capable of doing so sign delivery notifications. It further requires that delivery notifications that result from encrypted messages also be encrypted.

However, IMDN allows intermediary application servers to insert notification requests into messages, to add routing information to messages, and to act on notification requests. It also allows list servers to aggregate delivery notifications.

Such intermediaries will be unable to read end-to-end encrypted messages in order to interpret delivery notice requests. Intermediaries that insert information into end-to-end signed messages will cause the signature validation to fail. (See Section 9.1.)

10. Examples

The following sections show examples of S/MIME messages in SIP and MSRP. The examples include the tags "[start-hex]" and "[end-hex]" to denote binary content shown in hexadecimal. The tags are not part of the actual message, and do not count towards the Content-Length header field values.

In all of these examples, the clear text message is the string "Watson, come here - I want to see you." followed by a newline character.

The cast of characters includes Alice, with a SIP AoR of "alice@example.com", and Bob, with a SIP AoR of "bob@example.org".

Appendix A shows the detailed content of each S/MIME body.
10.1. Signed Message in SIP Including the Sender’s Certificate

Figure 1 shows a message signed by Alice. This body uses the "application/pcks7-mime" media type with an smime-type parameter value of "signed-data".

The S/MIME body includes Alice’s signing certificate. Even though the original message content is fairly short and only minimal SIP header fields are included, the total message size approaches the maximum allowed for the SIP MESSAGE method unless the UAC has advance knowledge that all SIP hops will use congestion-controlled transport protocols. A message that included all the SIP header fields that are commonly in use in some SIP deployments would likely exceed the limit.
Figure 1: Signed Message in SIP

10.2. Signed Message in SIP with No Certificate

Figure 2 shows the same message from Alice without the embedded certificate. The shorter total message length may be more manageable.
MESSAGE sip:bob@example.org SIP/2.0
Via: SIP/2.0/TCP alice-pc.example.com;branch=z9hG4bK776sgdkfie
Max-Forwards: 70
From: sip:alice@example.com;tag=49597
To: sip:bob@example.org
Call-ID: asd88asd66b@1.2.3.4
CSeq: 1 MESSAGE
Content-Transfer-Encoding: binary
Content-Type: application/pkcs7-mime; smime-type=signed-data;
            name="smime.p7m"
Content-Disposition: attachment; filename="smime.p7m"
Content-Length: 395

[start-hex]
3082018706092a8648886f70d010702a082017830820174020101310d300b0609
608648016503040201305306092a864886f70d010701a046044443f6e74656e
742d547970653a2a746578742f706c61696e0d0a0a576174736d6e2c20636f
6d65268648657069736b6f72696e747373652068657265202d206865207061
73746f6e7365742e0d0a31820701301c06092a864886f70d010905310f170d3139310d313030
313335345a302f6092a864886f70d0109043120420ef778fc94056e6dc2576
f47a599b3126195a9f1a227ada6f35fa22c050db195a300a6082a8648ce3d04
03020447304502203607275592d3607275592d30c85a931041a0180460c638ac9a8080918
87172a0887c8d4aa022100cd9e14bd21817336e9052fe509af2e2bcde16dd3e9
48d0f5f78a969e26382682
[end-hex]

Figure 2: Signed Message in SIP with No Certificate Included

10.3. MSRP Signed and Encrypted Message in a Single Chunk

Figure 3 shows a signed and encrypted message from Bob to Alice sent
via MSRP.

MSRP dsdfoe38sd SEND
To-Path: msrp://alicepc.example.org:7777/iau39soe2843z;tcp
From-Path: msrp://bobpc.example.org:8888/9d14eae923wzd;tcp
Message-ID: 456so39s
Byte-Range: 1-1940/1940
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/pkcs7-mime; smime-type=auth-enveloped-data;
            name="smime.p7m"

[start-hex]
30820790060b2a8648886f70d0109100117a082077f3082077b020103182024f
3082024b0201003033302631143012060355040a0c0b6578616d706c652e636f
30
6d310e300c06035504030c05416c69636502090083f50bb70bd5c40e300d0609
2a864886f70d010101050004820200759a61b4dfdf1f1af24686005635e476110
fa2723c1be9e45484b633e8387de967cb5c0eacf3b3551a56a1975cb550e7be31
c13d1da80b7f31204845b94ac6240424d9330561c843999415dd644b3c
ad95072f71451393c9f9282c4883bd0ccc5dd54b931464e0a08e5e55e92c51a68
de106521616c73ca8e764bb8ad789a87377675ef8dc3c6c0a6ed3ece8528cc6c6
a293d5095445191a1bcdf90e0f37e2c2eef0f4ec6225100cc062ec17489693bc
88b4e3d3f3c14703729dd5378e758accf31d86f2fa147e22c3c97a96c6b427
29aad27f37ae4ac98474d1eeb48948c12a403d0bce308a218da64f5692497c
0c596f4664df3eb5f31841158dfc3b84090aa06380aa065137ee16995ca1974167
9d7a3c90e766d5889bb5a467424e3b0b7d6f7f80cb4ab67bc343662a6
3f5e59f51149c535920caca2b08a138c227e6768ad9400f0df3d3ae1cbeceaa03
0dce2f61b00f5ceaa35dee1303cfbc9d8ee1cbfd6f29b9eb6eb24ad1d7bb8f580
f114a13b989094bf404a5ada7e764b7f8c5ff9294952b359e9693b9ad63c051
5c95cc6f823c2020106a22262413ef3e973df48f7b6143f8b7ae81a48ad3c0e
1abaa3cf39e7e3662005cca0611bf0ac00eeefa1498f2d259b90f7fda838ef6f1b
061f387c2c48c8b5ddacb8623083f23f7925165c95e9eb64779984981dd697
b447f40c407989bb890cc2e9590aff73808050f06092a864886f70d0107010301
06090684680160300010604004cd8757222ae5c294117f0fc120010108082
04e0fe2f3ed0bfe0f6998c39bf4a52f9abf80b0fee3d7e2e85181ae1c9891ea2e
decd9404885612dc6f984334db602bb7749b2504e45f573c3b066626b0fc746236
1eccc26750139be5c5d2696cf518522785e53c381cb0a68c5989de4f4c
e14a33384e4de55dfed787687027a31e4a7cc0299144c5d6e6a93e996079
0e50e7bf00dada0b8b36f4e3b321a4518381d9821a11a0b9343ac10c979b54ee
b73b38c131aafc5610e3734877474d9cafa9514902886c46f6962d2eb3f3f904
1aa4ee1b88d6c9858d35df65b08b52a9e352298858d0ae416b86f0a88d7bea7a9
81dbd283e8b94ed50b8f6265c2348a18a169aaacb5a37a529bda29cb10e0eddfc
14231095d87964673bd3f1b13c68b4cfff91a9069609c62a183d02375a15c5829
f3eaa38f24df62b31803773d7313f75db1841f9d985b653d4a462b2617126326
cf1cf833a547525325c8417a094484d7b64bc5bf126a9eb3a0ed94caf1dbd23
6990c654f7eb9c9e8952e2f6d068eef8b8a33bc64c6ddca7aef4d3574737d7c4c
1e9737d08bf3f22dea61d73083c324038ce3b1d33d3383a89875e241c2ed7cb6
80758c041069489860fc9940852360725483b246988f999953acfc16ec585b7aa
85e555c44970l04adb0439ed103d3c54df84b29cb04b8ce5d540c843487da79c186
5d65188f93fa9e49b8a40c70664b9026d607ba9c89e1984d8a099dbd1bf821fc
862510db59f949c3f0f9c6d7beff2793d184cad7ac131c1a5295f89ef3eb3
7ebbe02608a28c5a13878e6607ade4f5e8411590920ba878a79305a31562c
2229e42b886d60481c8473f9d51269e2a6f361bce207f868e607784ed46150e
04ff50cd20c95bc1275716f6e06bc4aa9a72d8f1fe4fc0fc8066d64b365fafa8e6
8c1b4e7a9721deacc16ca350a28eafa25fac054dd934bfe7e5f4af753a4a1956
8c7e9ec4b9e0a02c07b04874a0662d24284c09d9e8abe1f713726b2b46f511488e
933eda92e7d1774564963b3913cbdb9f1c27fefe36f832e05155fc39ee6552fa7
b4188975ec5c673b32c9f213c8ac6b8e132a5a7c3fb74f01605cd8e201d10521
93e168d4435de3887d321ba2f179293cfeeb9bbde271d26f56ab0e611c99c9
cdee2b88cd8377377eaeef37df85b7a7a2bbcaec75e2617de601c02b86e9d9a40f3
20462c56d6835176dcd0614bf3a060f75c0631c920845ed8c0b5add35df19f
84ff221cd3b529cd1028845f808954df4f1441de361bfb31a5f4fc8c2b708d
50b645d4e7db88648c3eef14765158fb0e8db3b53ddcbe26d7124c6e1992f8
10.4. MSRP Signed and Encrypted Message sent in Multiple Chunks

Figure 4 shows the same message as in Figure 3 except that the message is broken into two chunks. The S/MIME operations were performed prior to breaking the message into chunks.

MSRP d93kswow SEND
To-Path: msrp://alicepc.example.com:7777/iau39soe2843z;tcp
From-Path: msrp://bobpc.example.org:8888/9di4eae923wzd;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-960/1940
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
  name="smime.p7m"

[start-hex]
30820790060b2a864886f70d0109100117a082077f3082077b020100318204f
308204b0201003303233261143012060355040a0c06b6578616d656e656261
630e30c06035504030504166963653209083f50bb70bd5c04e0300d609
2a684886f70d0101010000482020759a61b4ddf11af12466800563e476110
fa2723c19e454848b6d3e8387de967dc50cafb35571a56a1975c5b50ebe31

c131da80f731024815450bb8d4cac2604042d9330561c8439999415dd6493c
ad95072f71451393c9f9282c883bd0ccc5dd54b931464e00a6e55e592c51a68
de1062156ec7d3ca8764bb8ac789a88377765ef8dc36c0a6ed3ecca5285cac6
a29d5059454191ab1dcbf9060ff37e2c2eef4e6c225100cc062e1c748963bdc
88b8e3dfcf714073729d5c7583e758acfc3d186f2fa417be22c37c9a76c6b47
29aad27f3ae44ac98474d1eeb48948c12a403d0b3ce08a2186daf456924987c
ce5c9666f6debf3f18114158dfc3bb4090aa60380aa865137e16995c51897167
9d7a3c9b0a79ed7d5c8d89bb54a66742e3a43b07d6f78bc0a4b67bc34362a62
35fe595f1149c35900402a31c8227e6f76a8d940400fd3d3ea1c8ee0a03

dceef2ffbf00f5ce335de1303cfcbf938e1cbf6d62f19be624ba3d1d7b8f580
f114a13b890894fb4044a5daa7647f8c5f92949452b35aeb9639b8ad63c051

[Page 20]
Internet-Draft          S/MIME for SIP Messaging            January 2019

5c95ccc6f823c2201067ea43f97d48f7b71634f82ae8e148cad3ae0
1abaa3cf9ee7e36320e05cca061fb0c00ef1a48f2d259bf6f7da83e6f61fb
061f387c2dc48c8b5dbaca862308f324f7925165c9e5ebb46779988491832g97
b447f4c40798b98b0c9e580af783082050f06092a8684886f70d10710310e
006096808405030401063011040c4d875722eac52594117f0c120201108082
04e0fe2fb3de0f06998c39b4a452f9bf8b0fe3d7e2e8518aeecfa89812de
decd9404885612dfc684334d8602s7749b2504e45f57c3b666260bf7c476236
1e2ee267c560139be526a2af9696cf51585272852c3818cab0a68598def4ec
14a33388e4e5d5f5ed77867027a31e4a7c0299144c5de6bae3969e70
0e057ebf0d9ad73b8369f4eb321b4153871d982a11a0b3943ac10c97b54be
b73b38ec131af5610e3734872747d69cafa591402886c64f962d242f33f904
1a4ae11b88dc6958d53df5b08bb2a5e3229885d0ae41686f08d0eb7a9
81dabc283e8b98e26626c2348a18a169aacc53a7529bd2af9b10efddcf
14231095d87964373db3f13c6b4cff9a1906960c1ea2301d325b7a15c5829
[end-hex]        -------d93kswow+

MSRP op2nc9a SEND
To-Path: msrp://alicepc.example.com:8888/9d41e9a293wzd;tcp
From-Path: msrp://bobpc.example.org:7654/iau39soe2843z;tcp
Message-ID: 12339sdqwer
Byte-Range: 961-1940/1940
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
        name="smime.p7m"

[start-hex]
f3ea038f24df6b23180377d37131f75db18f41f9d85b653da46bf2617126326
cfc1cb833457752352c8417a09484d7b64bcf51b26a9beb8aede4b9eaf1b2d3
c690c654f7eb9c9e985e2f6d6068fe8ba33bc64ddcda7ae4d35743737d7c4dc
1e93770df8f4f22eda61d73083c32c4038c1eb3dd3333a9a8795e241c2ed7cb
80758c04106948e606f9490e85236072548b32496989f99953acdefe5b7a8
85e554ac49701a6d4b039ed103dc458df4b29cb04b8cedd540c843484da79c18
56d518f89f3a9e4b9b84070664b90296c60b7ac98e9184d49809d9dbdfb281f
0b26510db5d9f97ad9c9f10f9c67dbe7f29391d184cad7ac131c16a5295f89fe3bb
7eb8e02085a828c5a138786607ade4ef5e8d4115909209ba87a9739a5316c2
2229e42b8686064b1c847f9df5d269a2ef6341bce20ff7688607784e6d415e0
04f50cd209c5b12751169fe086bc4a9a728df1ef4cfc0b96667a3ffaa686
8eb4f53e7a9212aecc1b6ca350a28eaf25fac054dd934bfe7e5f4af753a4a1596
87ebeb4349e0ac0270b4874a068d22484c09d9e8abe17f1372b4b2f6511f48e8
933ed92e5d1774564963b391c3b9df1c27ffe36f832e05155fc39ee6652fa7
b4188975ec5b7bc63f29213c8ac6b8e132a5a7c3bf74f016405cd8e201d5021
93e186bf4358e387d7321ab2f1792f3cfeeb9bbede7211d26f56ab06e11ccc9c
cdde2b88c83783773eac37f9d85b7a7a2bcaec752e617d60e10b2b6ed9a40f3
20462c5d6f8351716dcd6014bdf3a0f75fc6319c920845ed8c0bad35df19
84f2124035b26c1028845f0809543df4f144led36b1bf31a5af8c2b708d
50b645d4e7db88648c3eefe14765158f9b0e8d3bb53d3cbe267124c6e1df92f8
11.  IANA Considerations

This document makes no requests of the IANA.

12.  Security Considerations


The security considerations from algorithms recommended in this document also apply, see [RFC3565], [RFC5480], [RFC5753], [RFC5754], [RFC7748], [RFC8032], [RFC8419], and [RFC8418].

This document assumes that end-entity certificate validation is provided by a chain of trust to a certification authority (CA), using a public key infrastructure.  The security considerations from [RFC5280] apply.  However, other validations methods may be possible; for example sending a signed fingerprint for the end-entity in SDP.  The relationship of this work and the techniques discussed in [RFC8224] and [I-D.ietf-sipbrandy-rtpsec] are out of scope for this document.

When matching an end-entity certificate to the sender or recipient identity, the respective SIP AoRs are used.  Typically these will match the SIP From and To header fields.  Some UAs may extract sender identity from SIP AoRs in other header fields, for example, P-Asserted-Identity [RFC3325].  In general, the UAS should compare the certificate to the identity that it relies upon, for example for
display to the end-user or comparison against message filtering rules.

The secure notification use case discussed in Section 1 has significant vulnerabilities when used in an insecure environment. For example, "phishing" messages could be used to trick users into revealing credentials. Eavesdroppers could learn confirmation codes from unprotected two-factor authentication messages. Unsolicited messages sent by impersonators could tarnish the reputation of an organization. While hop-by-hop protection can mitigate some of those risks, it still leaves messages vulnerable to malicious or compromised intermediaries. End-to-end protection prevents modification by intermediaries. However, neither provide much protection unless the recipient knows to expect messages from a particular sender to be signed and refuses to accept unsigned messages that appear to be from that source.

Mobile messaging is typically an online application; online certificate revocation checks should usually be feasible.

S/MIME does not normally protect the SIP or MSRP headers. While it normally does protect the CPIM header, certain CPIM header fields may not be protected if the sender excludes them from the encrypted or signed part of the message. (See Section 9.1.) Certain messaging services, for example those based on RCS, may include intermediaries that attach metadata to user generated messages in the form of SIP, MSRP, or CPIM header fields. This metadata could possibly reveal information to third parties that the sender might prefer not to send as cleartext. Implementors and operators should consider whether inserted metadata may create privacy leaks. Such an analysis is beyond the scope of this document.

MSRP messages broken into chunks must be reassembled by the recipient prior to decrypting or validation of signatures (see Section 8.1). Section 14.5 of [RFC4975] describes a potential denial of service attack where the attacker puts large values in the byte-range header field. Implementations should sanity check these values before allocating memory space for reassembly.

Modification of the ciphertext in EnvelopedData can go undetected if authentication is not also used, which is the case when sending EnvelopedData without wrapping it in SignedData or enclosing SignedData within it. This is one of the reasons for moving from EnvelopedData to AuthEnvelopedData, as the authenticated encryption algorithms provide the authentication without needing the SignedData layer.
An attack on S/MIME implementations of HTML and multipart/mixed messages is highlighted in [Efail]. To avoid this attack, clients MUST ensure that a text/html content type is a complete HTML document. Clients SHOULD treat each of the different pieces of the multipart/mixed construct as coming from different origins. Clients MUST treat each encrypted or signed piece of a MIME message as being from different origins both from unprotected content and from each other.

13. References

13.1. Normative References

[I-D.ietf-lamps-rfc5750-bis]

[I-D.ietf-lamps-rfc5751-bis]


13.2. Informative References


Internet-Draft          S/MIME for SIP Messaging            January 2019

Extensions to the Session Initiation Protocol (SIP) for
Asserted Identity within Trusted Networks", RFC 3325,
DOI 10.17487/RFC3325, November 2002,

[ RFC 3840 ] Rosenberg, J., Schulzrinne, H., and P. Kyzivat,
"Indicating User Agent Capabilities in the Session
Initiation Protocol (SIP)", RFC 3840,
DOI 10.17487/RFC3840, August 2004,

[ RFC 3860 ] Peterson, J., "Common Profile for Instant Messaging
(CPIM)", RFC 3860, DOI 10.17487/RFC3860, August 2004,

Messaging (CPIM): Message Format", RFC 3862,
DOI 10.17487/RFC3862, August 2004,

[ RFC 4648 ] Josefsson, S., "The Base16, Base32, and Base64 Data
Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,

for the Message Sessions Relay Protocol (MSRP)", RFC 4976,
DOI 10.17487/RFC4976, September 2007,

[ RFC 5438 ] Burger, E. and H. Khartabil, "Instant Message Disposition
Notification (IMDN)", RFC 5438, DOI 10.17487/RFC5438,

RFC 6121, DOI 10.17487/RFC6121, March 2011,

[ RFC 7515 ] Jones, M., Bradley, J., and N. Sakimura, "JSON Web
Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May

RFC 7516, DOI 10.17487/RFC7516, May 2015,
Appendix A.  Message Details

The following section shows the detailed content of the S/MIME bodies used in Section 10.

A.1.  Signed Message

Figure 5 shows the details of the message signed by Alice used in the example in Section 10.1.

CMS_ContentInfo:
  contentType: pkcs7-signedData (1.2.840.113549.1.7.2)
  d.signedData:
    version: 1
    digestAlgorithms:
      algorithm: sha256 (2.16.840.1.101.3.4.2.1)
      parameter: <ABSENT>
    encapContentInfo:
      eContentType: pkcs7-data (1.2.840.113549.1.7.1)
      eContent:
        0000 - 43 6f 6e 74 65 6e 74 2d-54 79 70 65 3a 20 74    Content-Type: t
        000f - 65 78 74 2f 70 6c 61 69-6e 0d 0a 0d 0a 57 61    ext/plain....Wa
        001e - 74 73 6f 6e 2c 20 63 6f-6d 57 61 74 73 6f 6e, come here
        002d - 20 2d 20 49 20 77 61 74-20 73 65 20 73 65 73 65 - I want to see you...
        003c - 65 20 79 75 2f 75 2e 0d 0a- e you...
          certificates:
            d.certificate:
              cert_info:
version: 2
serialNumber: 13292724773353297200
signature:
  algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
  parameter: <ABSENT>
issuer: O=example.com, CN=Alice
validity:
  notBefore: Dec 19 23:12:05 2017 GMT
  notAfter: Dec 19 23:12:05 2018 GMT
subject: O=example.com, CN=Alice
key:
  algor:
    algorithm: id-ecPublicKey (1.2.840.10045.2.1)
    parameter: OBJECT:prime256v1 (1.2.840.10045.3.1.7)
  public_key: 0000 - 04 d8 7b 54 72 9f 2c 22-fe eb d9 dd ba 0e ..(Tr.,"
000e - 40 64 22 97 a6 09 38-87 a4 da e7 99 0b .0d"...8......
001c - 23 f8 7f a7 ed 99 db 8c-f5 a3 14 f2 ee 64 #............d
002a - 10 6e f1 ed 61 db fc 0a-4b 91 c9 53 cb d0 .n...a...K..S...
0038 - 22 a7 51 b9 14 80 7b b7-94 ".Q...{
issuerUID: <ABSENT>
sujectUID: <ABSENT>
extensions:
  object: X509v3 Subject Alternative Name (2.5.29.17)
critical: BOOL ABSENT
value:
  0000 - 30 17 86 15 73 69 70 3a-61 6c 69 63 65 0...sip:alice
  000d - 40 65 78 61 6d 70 6c-65 2e 63 6f 6d @example.com
sig_alg:
  algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
  parameter: <ABSENT>
signature: (0 unused bits)
  0000 - 30 45 02 20 78 79 be 1c-27 f8 46 27 6f df 15 .OE. xy.'F'o..
  000f - e3 33 e5 3c 6f 17 a7 57-38 8a 02 cb 7b 8a e4 .3.<o..W8...{
  001e - 81 c1 64 1a e7 a9 02 21-00 ff 99 cd 9c 94 07 ..d...........
  002d - 6c 82 b0 2f ea 3b 13 50-17 9a 4b 77 52 e1 6f 1.../;,.P..KwR.o
  003c - a3 0a 3f 9a b2 96 50 50-e2 81 89 ...?...P....
crls:
  <ABSENT>
signerInfos:
  version: 1
d.issuerAndSerialNumber:
    issuer: O=example.com, CN=Alice
    serialNumber: 13292724773353297200
digestAlgorithm:
  algorithm: sha256 (2.16.840.1.101.3.4.2.1)
  parameter: <ABSENT>
signedAttrs:
object: contentType (1.2.840.113549.1.9.3)
set:
   OBJECT:pkcs7-data (1.2.840.113549.1.7.1)

object: signingTime (1.2.840.113549.1.9.5)
set:
   UTCTIME: Jan 24 23:52:56 2019 GMT

object: messageDigest (1.2.840.113549.1.9.4)
set:
   OCTET STRING:
      0000 - ef 77 8f c9 40 d5 e6 dc-25 76 f4 7a 59 \..0...v.zY
      000d - 9b 31 26 19 5a 9f 1a 22-7a da f3 5f a2 .1&.Z.."z.._
      001a - 2c 05 0d 8d 19 5a                        ,....Z
      signatureAlgorithm:
         algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
         parameter: <ABSENT>
      signature:
         0000 - 30 45 02 20 58 79 cc 62-85 e0 86 06 19 d3 b6 OE. Xy.b.......n.
         000f - 53 d4 67 9f 03 73 d7 45-20 cf 56 10 c2 55 5b S.g..s.E .V..U[
         001e - 7b ec 61 d4 72 dc 02 21-00 83 aa 53 44 28 4d {.a...!....SD(M
         002d - 4c ef de 31 07 9c f9 71-7b 6d 6e c8 71 e9 L..l...q.i]n.q.
         003c - a4 60 ec 2e 12 65 2b 77-a4 62 4d ."..e+w.bM
      unsignedAttrs:
         <ABSENT>

Figure 5: Signed Message

A.2. Short Signed Message

Figure 6 shows the message signed by Alice with no imbedded certificate, as used in the example in Section 10.2.

CMS_ContentInfo:
   contentType: pkcs7-signedData (1.2.840.113549.1.7.2)
   d.signedData:
      version: 1
      digestAlgorithms:
         algorithm: sha256 (2.16.840.1.101.3.4.2.1)
         parameter: <ABSENT>
      encapContentInfo:
         eContentType: pkcs7-data (1.2.840.113549.1.7.1)
         eContent:
            0000 - 43 6f 6e 74-7a 70 65 3a 20 74 Content-Type: t ext/plain....Wa
            000f - 65 78 74 2f 70 6c 61 69-6e 0d 0a 0d 0a 57 61 tson, come here
            001e - 74 73 6f 6e 2c 20 20 63-20 77 61 74 20 74 e you...
            002d - 20 2d 20 49 20 77 61 6e-74 20 74 6f 20 73 65 - I want to se
            003c - 65 20 79 74 6f 75 2e 0d Content-Type: t e you...

Campbell & Housley  Expires July 30, 2019
certificates:
<ABSENT>
crls:
<ABSENT>
signerInfos:
  version: 1
d.issuerAndSerialNumber:
    issuer: O=example.com, CN=Alice
    serialNumber: 13292724773353297200
  digestAlgorithm:
  algorithm: sha256 (2.16.840.1.101.3.4.2.1)
  parameter: <ABSENT>
signedAttrs:
  object: contentType (1.2.840.113549.1.9.3)
  set:
    OBJECT:pkcs7-data (1.2.840.113549.1.7.1)
  object: signingTime (1.2.840.113549.1.9.5)
  set:
    UTCTIME: Jan 24 23:52:56 2019 GMT
  object: messageDigest (1.2.840.113549.1.9.4)
  set:
    OCTET STRING:
    0000 - ef 77 8f c9 40 d5 e6 dc-25 76 f4 7a 59 .w..@...%v.zY
    000d - 9b 31 26 19 5a 9f 1a 22-7a da f3 5f a2 .1&.Z.."z..__.
    001a - 2c 05 0d 8d 19 5a                        ,....Z
  signatureAlgorithm:
  algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
  parameter: <ABSENT>
signature:
  0000 - 30 44 02 20 1c 51 6e ed-9c 10 10 a2 87 e1 11 0D. .Qn........
  000f - 6b af 76 1d f1 c4 e6 48-da ea 17 89 bc e2 8a k.v.....H.......
  001e - 9d 8a f4 a4 ae f9 02 20-72 7f 5e 4b cc e2 0b ........ r.^K...
  002d - cf 3c af 07 c8 1c 11 64-f0 21 e7 70 e0 f6 a0 <.....d!.p....
  003c - 96 2e 0a 7b 19 b7 42 ad-cb 34 ...{..B..4
unsignedAttrs:
<ABSENT>
Figure 6: Signed Message without Imbedded Certificate

A.3. Signed and Encrypted Message

The following sections show details for the message signed by Bob and encrypted to Alice, as used in the examples in Section 10.3 and Section 10.4.
A.3.1. Signed Message Prior to Encryption

CMS_ContentInfo:
  contentType: pkcs7-signedData (1.2.840.113549.1.7.2)
  d.signedData:
    version: 1
    digestAlgorithms:
      algorithm: sha256 (2.16.840.1.101.3.4.2.1)
      parameter: <ABSENT>
    encapContentInfo:
      eContentType: pkcs7-data (1.2.840.113549.1.7.1)
      eContent:
        0000 - 43 6f 6e 74 65 6e 74 2d-54 79 70 65 3a 20 74   Content-Type: t
        000f - 65 78 74 2f 70 6c 61 69-6e 0d 0a 0d 0a 57 61   ext/plain....Wa
        001e - 74 73 6f 6e 2c 20 63 6f-6d 65 20 68 65 72 65   tson, come here
        002d - 20 2d 20 49 20 77 61 6e-74 20 74 6f 20 73 65    - I want to se
        003c - 65 20 79 6f 75 2e 0d 0a- e you...
      certificates:
        d.certificate:
          cert_info:
            version: 2
            serialNumber: 11914627415941064473
            signature:
              algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
              parameter: <ABSENT>
            issuer: O=example.org, CN=Bob
            validity:
              notBefore: Dec 20 23:07:49 2017 GMT
              notAfter: Dec 20 23:07:49 2018 GMT
            subject: O=example.org, CN=Bob
            key:
              algor:
                algorithm: id-ecPublicKey (1.2.840.10045.2.1)
                parameter: OBJECT:prime256v1 (1.2.840.10045.3.1.7)
              public_key: (0 unused bits)
            issuerUID: <ABSENT>
            subjectUID: <ABSENT>
            extensions:
              object: X509v3 Subject Alternative Name (2.5.29.17)
              critical: TRUE
              value:
                0000 - 30 15 86 13 73 69 70 70 3a-62 6f 62 40 65 0...sip:bob@e
                000d - 78 61 6d 70 6c 65 2e 6f-72 67 67 xample.org
sig_alg:
   algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
   parameter: <ABSENT>
   signature: (0 unused bits)
0000 - 30 45 02 21 00 b2 24 8c-92 40 28 22 38 9e c9  0E.!!$.@(*8.
000f - 25 7f 64 cf fd 10 6f ba-0b 96 c1 19 07 30 34  %.d...o......04
001e - d5 1b 10 2f 73 39 6c 02-20 15 8e b1 51 f0 85 .../s9l....Q..
002d - b9 bd 2e 04 cf 27 8f 0d-52 2e 6b b6 fe 4f 36 .....’..R.k..06
003c - f7 4c 77 10 b1 5a 4f 47-9d e4 0d  .Lw...ZOG...
crls:
   <ABSENT>
signerInfos:
   version: 1
d.issuerAndSerialNumber:
   issuer: O=example.org, CN=Bob
   serialNumber: 11914627415941064473
digestAlgorithm:
   algorithm: sha256 (2.16.840.1.101.3.4.2.1)
   parameter: <ABSENT>
signedAttrs:
   object: contentType (1.2.840.113549.1.9.3)
   set:
      OBJECT:pkcs7-data (1.2.840.113549.1.7.1)
   object: signingTime (1.2.840.113549.1.9.5)
   set:
      UTCTIME:Jan 24 23:52:56 2019 GMT
   object: messageDigest (1.2.840.113549.1.9.4)
   set:
      OCTET STRING:
0000 - ef 77 8f c9 40 d5 e6 dc-25 76 f4 7a 59  .w..@...v.zY
000d - 9b 31 26 19 5a 9f 1a 22-7a da f3 5f a2  .1&.Z..'z.._
001a - 2c 05 0d 8d 19 5a ,....Z
   signatureAlgorithm:
   algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
   parameter: <ABSENT>
   signature:
0000 - 30 45 02 21 00 f7 88 ed-44 6a b7 0f ff 2c 1f 0E.!....Dj....
000f - fa 4c 03 74 fd 08 77 fd-61 ee 91 7c 31 45 b3 .L.t...w.a..|E.
001e - 89 a6 76 15 c7 46 fa 02-20 77 94 ad c5 7f 00 ..v..F.. w.....
002d - 61 c7 84 b9 61 23 cc 6e-54 bb 82 82 65 b6 d4 a...a#.nT....e..
003c - cc 12 99 76 a6 b1 fc 6d-bc 28 d6 ...v...m饴
unsignedAttrs:
   <ABSENT>

Figure 7: Message Signed by Bob prior to Encryption
A.3.2. Encrypted Message

CMS_ContentInfo:
  contentType: pkcs7-authEnvelopedData (1.2.840.113549.1.9.16.1.23)
  d.authEnvelopedData:
    version: 0
    originatorInfo: <ABSENT>
    recipientInfos:
      d.ktri:
        version: <ABSENT>
        d.issuerAndSerialNumber:
          issuer: O=example.com, CN=Alice
          serialNumber: 9508519069068149774
        keyEncryptionAlgorithm:
          algorithm: rsaEncryption (1.2.840.113549.1.1.1)
        parameter: NULL
    encryptedKey:
      0000 - 75 9a 61 b4 dd f1 f1 af-24 66 80 05 63 5e 47 .u.a.....$f..c^G
      000f - 61 10 fa 27 23 c1 b9 e4-54 84 b6 d3 3e 83 87 a~...'...T...>
      001e - de 96 7d c5 e0 ca fb 35-57 1a 56 a1 97 5c b5 ...).5W.V..
      002d - 50 e7 be 31 c1 31 da 80-fb 73 10 24 84 5b ab P..1...s.].
      003c - b8 d6 4c ac 26 04 04 24-d9 33 05 61 c8 43 99 ..L...$.3.a.C.
      004b - 94 15 dd 64 4b 3c ad 95-07 2f 71 45 13 93 c9 ...dK<./qE.
      005a - 9f 28 2c 48 83 bd 0c cc-5d 4b 93 14 64 ea0 ,(.H...).K..d.
      0069 - 0a 6e 55 e5 92 c5 1a 6b-de 10 62 51 6e c7 d3 .nU.....h..bQn.
      0078 - ca 8e 76 4b b8 ac 78 7a-9a-88 37 77 65 ef 8d c3 ..vk..7we.
      0087 - 6c 0a 6e d3 ec ae 52 85-ca c6 a2 9d 50 59 44 l.n...R....PYD
      0096 - 57 19 a1 bd cf 90 6e 0f-f3 7e 2c 2e f0 4c ec W.....n...~.
      00a5 - 62 25 10 0c c0 62 e1 c7-48 96 3b bc 88 b8 e3 b%...b.H;....
      00b4 - df cf 71 40 73 72 9d d5-c7 58 3e 75 8a cf 3d ..q@sr...x.u=.
      00c3 - 18 6f 2f a4 17 be 22 c3-7c 9a 76 c6 b4 27 29 .o/...".v...'}
      00d2 - aa d2 7f 73 ae 44 ac 98-47 4d 1e eb 48 94 8c ...s.D..GM..H..
      00e1 - 12 a4 03 d0 b3 ce 08 a2-18 d6 af 45 69 24 89 ..........Ei$.
      00f0 - 7c c5 c9 66 4f 6d fe b3-f1 81 41 15 8d fc 3b |..fOme...A;...
      00ff - 84 09 0a a6 03 80 aa 86-51 37 e1 69 9c 5c 81 .........Q7.i.\.
      010e - 97 41 67 9d 7a 3c 9a ba-79 e6 d7 d5 c8 d8 9b .Ag.z<...y....
      011d - b5 4a 66 74 23 e4 3b 0b-7d 6f 78 c0 ab 67 .Jft#.;}ox...g
      012c - bc 34 36 62 a6 35 fe 59-5f 11 49 c5 39 50 ca ..46b.5.Y_I.9P.
      013b - c2 e0 ba 31 8c 22 7e 6f-76 a8 d9 40 0f 0d 3d ...1"-ov..@8..
      014a - d3 ea 1c 8e ce a0 03 dc-ce 2f 1f b0 0f 5c ea .........../\.
      0159 - 33 5d e5 30 3f cb f9 3d-8e 1c bf d6 82 f1 9b 3].0?"....
      0168 - eb 62 4b ac d1 d7 b8 f5-80 f1 14 a1 3b 89 08 .bK...............
authEncryptedContentInfo:
  contentType: pkcs7-data (1.2.840.113549.1.7.1)
  contentEncryptionAlgorithm:
    algorithm: aes-128-gcm (2.16.840.1.101.3.4.1.6)
    parameter:
      aes-nonce:
        0000 - 4d 87 57 22 2e ac 52 94-11 7f 0c 12 M.W"..R.....
      aes-ICVlen: 16
  encryptedContent:
    0000 - fe 2f b3 de 0b f0 69 98-c3 9b f4 a9 52 fa bf ./.i.i...R..
    000f - 80 0f ef ee 3d 7e 2e 85 18-1a ec f1 a8 9e 1a 2e ...="~...
    001e - de cd 94 04 88 56 12 df-c6 98 43 34 d8 60 2b .....V.....C4.
    002d - 77 49 b2 50 4e 45 f5 7c-06 66 26 b0 fc 74 wI.PNE.[f&.t
    003c - 62 36 1e ec 26 7c 56 01-39 be 5c d2 86 a2 af b6.6&.V.9.
    004b - 96 9f cf 51 22 78 e5-2c 38 18 ca b0 a6 8c Q."x.8.....
    005a - 59 8d e4 fc e1 4a 33 3b-84 e4 de 5d df 57 ed Y...J38...].W.
    0069 - d7 88 67 02 7a 31 e4 a7-c0 29 91 44 c5 de ...g.z...].D.
    0078 - 6b ae 39 69 9e 70 05-7e b0 f0 da d7 3b 8b k.9.i.p........
    0087 - 36 9f 42 eb 32 1b 43-58 81 d9 82 a1 1a 0b 6.B.2.AS....
    0096 - 39 43 ac 10 c7 8b 5e-be 3b 38 13 1a fc 9C...(T.8...
04dd - 3e a8 db
   authAttrs:
      <EMPTY>
   mac:
     0000 - f6 ff c6 ae f1 9c d2-3d 98 5a 92 19 76 35
     000f - 2d
   unauthAttrs:
      <EMPTY>

Figure 8

Authors’ Addresses

Ben Campbell
Standard Velocity
204 Touchdown Dr
Irving, TX 75063
US

Email: ben@nostrum.com

Russ Housley
Vigil Security
516 Dranesville Rd
Herndon, VA 20170
US

Email: housley@vigilsec.com