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:

    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:

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

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

    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:
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)
    csor(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 UAs, 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/pkcs7-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.
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: 762

[start-hex]
30820f60609a864886f70d010702a08202e7308202e3020101310d300b0609
60864801650304020130530609a864886f70d010701a040644436fe674656e
742d547970653a20746578742f706cc6196ee0d0a0da5761747366e62c02636f
6d65206e5765202d200927616e7420746578742f0207656520796f752ed0da0a82
016b308201673082010da003020102020900b8793ec0e4c21530300a06082a86
48ce3d04030203631430120610350400a0c0b6578616d706c652e636f6d301e
300c06035504030c05416c696365301e170d3137313231393333323035353a
0d3138313231393233313230355a0a30263114301206505040a0c0b6578616d
6c652e63666d3010e300c06035504030c05416c6963653010300a6072a8648ce
3d020106082a8648ce3d0301703420004d8b54729f222feebd9ddba0e6a40
642297a6093887a4daee7790b23f87fa7ed99db8c5a314f2ee64106ef1ed61db
fc0a4b91c953cb022a751b914807bb794a3243022320063551d104193017
86157367973a616c696365040578616d706c652e636f6d300a06082a8648ce3d
0403020348003045020789be1c27f846276df15e333e53c6f17a575388a02
cb7b8ae481c1641ae7a9022100ff99cd9c9407682b02fe3b1350179a4b7752
e16fa30a3f9ab29650b0e2818931821093082010505020103330326314312
060355040a0c0b6578616d706c652e636f6d310e300c06035504030c05416c69
636520900b8793ec0e4c2153009b06096848016503040201a0693018069
2a864886f70d01090310b0609a864886f70d010701301c06902a864886f70d0
01090310f170d31393031323603631333545a320f06092a864886f70d0109
043122040f778fc94056dc2576f47a599b31261959f1a227adaf35fa22c
050d8d195a300a6082a8648ce3d04030204730450202005d2c5b5b04ff444a46
be468dfc7ef3b7de30019e4f052a223852189b35bb4e02210090e43a9d9846
cf2af8159c5c0e4f8488f2f39f998b1bb9b52a6fc6c776f2c8
[end-hex]

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.
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.com:7777/iau39soe2843z;tcp
From-Path: msrp://bobpc.example.org:8888/9di4eae923wzd;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]
308207906092a864886f70d01091002a082017830820174020101310d300b0609
608648016030420130506092a864886f70d010701a046044436f6e74656e
742d547970653a2a746578744f706c616669656e7469636f6a2c74686520
69830820150201301333326311430120355040a0c0b6578666666666666
63465d310e3006035504030c05416c69636520900b8793ec0e4c21530300b
06096086480160304201a693018d092a864886f70d010903310b0d092a86
4886f70d010710301c06092a864886f70d010905100f170d313930331323333
313335345a302f06092a864886f70d0109043122020ef778fc9405e6dc2576
f47a599b3126195a9f1a227dada935fa22c050d8d95a300a06082a8648ce3d04
030204473045020306207375592d30c8c5a931041a01804d60c638ac9a8080918
87172a0887c8d4aa022100cd9e14bd21817336e9052fe590af2e2bcde16dd3e9
48d0f5f78a969e26382682
[end-hex]
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]
30820790060b2a864886f70d0109100117a082077f3082077b0201003182024f
3082004b0201003303032631143012060355040a0c0b6578616d650e76e636f
6d310e0300635504030514061966563650209083f50bb70bd5c40e300d0609
2a864886f70d0101050004820200759a61b44d5f1af24668005635e476110
fa2723c9be45486bd3e3837de967dc5e0cafb3557a15a1975cbb50ebe31
b313da80b731024845bab8d64ac26040424d9305061c843999415dd44b3c
ad5072f71451398c9f282c4883d0cc5d5b99314640e006e55e592c51a68
de1f2516ce7d3ca8e764bb8ac789a8837775ef8dc36c0a6ed3e285cace6
a29d505945719a1bfc9060ff372e2ef0f4ec6225100cc02e1c748963bbc
88b8e3dfcf714073729dd5c7583e758af3d186f2fa417be223c79a76c6b47
29aadd27f73ae44ac98474d1eeb48948c12a4030b3ce08a218d6af456924897c
b5c59664f6debf3b18114158dfc3b84090aa6080aa865137e16995c518974167
9d7a3c9b0a796ed7d5c8d89b54a66742e43b07d6f78c0bab67bc343662a6
35e5f951149c5390cac2e0a318c227e6f76a8d94040f3d3ealce8ee003
dcece2f1f0b05fca335de1303fcbf93d8e1cbf682f19be6b24bacd1d7b8f580
f114al3b8f90894f4044a5daa764b7f8c5ff92949542b355e69369b8ad63c051

Figure 3: Signed and Encrypted Message in MSRP
Internet-Draft          S/MIME for SIP Messaging            January 2019

5c95ccc6f823c2201067ea43f97d48f7b6143f842ae8e1a48cad3ae0
1abaa3cf9ee7e36620e05ccaa61bfac00eeef1a498f2d259b9f0f7da83ef661b
061f387c7c2dc48c8b5dbaca862308f32f47925165c9e5ebb467799884918dd697
b447f4c4079f9b8b900c2e9580af783082050f0690a868488f6f70d1601301e
0609608640165030401603011040c4d875722aeac5294117f0c120201108082
04e0fe2fbd0ebf06998c39bf4ea952f4abf8b0fee3d7e2e85181aeecf1a89e1a2e
dec940488512dcfc694334d8602b7749b2504e45f573c3b0666260b7f476236
1ee26c7c56019b5e262a2af9696cf51852278e52c3818cab0a68c598ede4fc
1e14a33884e4de5df57edd78867027a31e4a7c0299144c5de66bae3969e970
0e057eb0f0dadc73b8369f4eb32bb14153871d982a11a0b3943ac10c97b54ee
b73b38ec131aafc5610e373487274de69caafa591402886c64f6962d2e23bf9f04
1a4ae1b1b88dc6958d53df0b8b52a3ae5229889850d6ae416b86f088d0eb79a
81dcd283e8b98e2c32484a18a169aacb5a37a529b2df9106ffedcfc
14231095d87964637bd33fb13c68b4cff91a1906960c1ea2301325b7a15c5829
[end-hex]        -------d93kswow+

MSRP op2nc9a SEND
To-Path: msrp://alicepc.example.com:8888/9di4eae923wzd;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]
f3ea03f24d6b23180377d37131f75db1b841f9d85b653da46bf2617126326
ccf1cb8334577525325c8417a094484d7b64bcf51b26a9e6e0d4b9caf1bd23
c90cc654f7eb9ce9852e2f6d068eef8ba33bc6c4ddca7afe4d357437d7c4dc
1e93770dd8f42f22ea61d73083c32c4038ce1b3d3383a89a6879e241c2ed7cb6
80758c04106948960fc9f490ae583260725488b324969f899953acfd6ec58b7a8
85e554ac449710a6d4b039ed103dc458df4b29cb04b8cedd540c843484da79c186
56d518f8ff9a9e49b84c070664b90296c0b7a9c8e9189d48a09dbd7b81fca
826510db59f9ad9c93f10f9c67d7ebf72931d841cad7a13c1a5295fc89fe3bb
7ebe082085a828c5a138786e607adef4f5e8d411590290ba87a79305a5316c2
2229e42b866d06481c8473f9df5126992aef6314bec02f76f8e8607784ed46150e
04f50cd209c5b127511369e6f06bc4aa9a72df8f1ef4fc0f86d66b365f8a86e
8c1bf43e7a9212aeec16ca350a28eafe25fac054d934bfe7e5f4af753a41596
8c7ebe4349e0ac0270b4874a068d22484c09d9e8abe17f1372b4b2f65f114e88
933eda92ed5174564963b391c3bdd9ffec2fffe36f832e05155fc39ee6652fa7
b4188975ec5b67c292f13c8ac6b8e132a5a7c3b7f4016405cd8c201d10521
93e1b6844358de38873211ba2f1792f3cfeeb9bbde7211d6265ab06e11ccc9c
cde2b888c8373773eaf37f8d5b7a7a2bbace752e6176de0c02b8e6d9a40f3
20462c5d6f8351716dcdc6014bd0f3a60f75fc0631c920845ed8c0bad35df19
84f22a10db3b529cd1028845f8089543df4f141ede36b1f3a5afcf8c2b708d
50b645d4e7db88648c3eeef14765158fb0e83db53dcbdce267124c6e1d92f8
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


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 65 20 68 65 72 65   tsone, come here
      002d - 20 2d 20 49 20 77 61 6e-74 20 73 65 2e 0d 0a  - I want to se
      003c - 65 20 79 75 2e 0d 0a-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: (0 unused bits)
  0000 - 04 d8 7b 54 72 9f 2c 22-fe eb d9 dd ba 0e ..(Tr.,"......
  000e - fa 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>
subjectUID: <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
  00d - 40 65 78 61 6d 70-2e 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 0E. 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 6a 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 0-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:
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 65 6e 74 2d-54 79 70 65 3a 20 74 Content-Type: t
                000f - 65 78 74 74 2f 70 6c 61 69-6e 0d 0a 0d 0a 0d 0a 57 ext/plain....Wa
                001e - 74 73 6f 6e 2c 20 63 6f-6d 20 63 57 6e 74 74 6f 70 son, come here
                002d - 20 2d 20 49 20 77 73 6f-20 77 61 6e 74 20 74 6f 73 - I want to se
                003c - 65 20 79 6f 75 2e 0d 0a 0a e you...

Campbell & Housley        Expires July 30, 2019                [Page 30]
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  OD. .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:
              alg:
                algorithm: id-ecPublicKey (1.2.840.10045.2.1)
                parameter: OBJECT:prime256v1 (1.2.840.10045.3.1.7)
              public_key: (0 unused bits)
              0000 - 04 86 4f ff fc 53 f1 a8-76 ca 69 b1 7e 27 ..O..S..v.i.-'
              000e - 48 7a 07 9c 71 52 ae 1b-13 7e 39 3b af 1a Hz...qR...-9;
              001c - ae bd 12 74 3c 7d 41 43-a2 fd 8a 37 0f 02 ...t<)AC...7..
              002a - ba 9d 03 b7 30 1f 1d a6-4e 30 55 94 bb 6f ....0...NOU...o
              0038 - 95 cb 71 fa 48 b6 d0 a3-83 ..q.H....
          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 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.!..$..(@"..!
  000f - 25 7f 64 cc 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..ZO...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..|1E.
  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.{...
    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.S.
003c - b8 d6 4c ac 26 04 04 24-d9 33 05 61 c8 43 99 ..L.&.$3.A.C.
004b - 94 15 dd 64 4b 3c ad 95-07 2f 71 45 13 93 c9 ...dK</.qE.
005a - 9f 28 2c 48 83 bd 0c cc-5d 54 b9 93 14 6b e0 ...
0069 - 0a 6e 55 e5 92 c5 1a 6b-de 10 62 51 6e c7 d3 .nU....h.bqN.
0078 - ca 8e 76 4b b8 ac 78 9a-88 37 77 65 ef 8d c3 ...
0087 - 6c 0a 6e d3 ec ae 52 85-ca c6 a2 9d 50 59 44 1.n...R.PYD
0096 - 57 19 a1 bd cf 90 6e 0f-f3 7e 2c 2e f0 f4 ec W....n...~
00a5 - 62 25 10 0c c0 62 e1 c7-48 96 3b bc 88 b8 e3 b%...b.H.;
00b4 - df cf 71 40 73 72 9d 05-c7 58 3e 75 8a cf 3d ..q@sr...Xu...=
00c3 - 18 6f 2f a4 17 be 22 c3-7c 9a 76 c6 b4 27 29 .o/...".v'.
00d2 - aa d2 7f 73 ae 44 ac 98-47 4d 1e eb 48 94 8c ...
00e1 - 12 a4 03 d0 b3 ce 08 a2-18 d6 af 45 69 24 89 ........Ei$
00ff - 7c c5 c9 6e 5f 6d fe b3-f1 81 41 15 8d fc 3b |...Fom...A;
00ff - 84 09 0a a6 03 80 aa 86-51 3e 69 7d 5c b1 ........Q7.i.\.
010e - 97 41 67 9d 7a 3c 90 ba-79 e6 d7 d5 c8 8d 9b .Ag.z<y......
011d - b5 4a 66 74 23 e4 3b 0b-7d 6f 78 c0 b4 ab 67 .Jft#;.)ox...g
012c - bc 34 36 62 a6 35 fe 59-5f 4f 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 d3 ...1."-ov...8@
014a - d3 ea 1c 8e ce a0 03 dc-ce 2f 1f b0 0f 5c ea ...........
0159 - 33 5d e1 30 3f cb f9 3d-8e 1c bf d6 82 f1 9b 3].0?.....
0168 - eb 62 4b ac d1 7b b8 50-f1 14 a1 3b 89 08 .bK...........
0177 - 94 fb 40 44 a5 da a7 64-b7 f8 c5 ff 92 94 94 ...@D...d.....
0186 - 52 b3 5a eb 96 39 b8 ad-63 c0 51 5c 95 cc c6 R..z...cQ.
0195 - f8 23 c2 20 10 67 ea 22-62 41 3f ef 39 7d 48 .#...g."bA?.9)H
01a4 - f7 b6 14 3f 84 2a e8 e1-a4 8c ad 3a e0 1a ba .....?........
01b3 - a3 cf 9e e7 e3 66 20 e0-5c ca 06 11 bf ac 00 .....f .\......
01c2 - ee f1 a4 98 f2 d2 59 b9-f0 f7 da 83 ef 6f 1b ......Y......o.
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.....

encryptedContent:
  0000 - fe 2f b3 de 0b f0 69 98-c3 9b f4 a9 52 fa bf ./.i....R..
  000f - 8b 0f ee 3d 7e 2e 85 18-la ec f1 a8 9e 1a 2e ....="....
  001e - de cd 94 04 88 56 12 df-c6 98 43 34 d8 60 2b .....V.C4.+n
  002d - 77 49 52 4e 45 f5 7c-3b 06 66 26 b0 fc 74 wI.PNE.|.f&
  003c - 62 36 1e ec 26 7c 56 01-39 be 5c d2 86 a2 af b6.&|V.9..
  004b - 96 4f 5f 51 85 22 78 e5-2c 18 ca b0 a6 8c Q."X,......
  005a - 59 8d e4 fc e1 4a 33 38-84 e4 de 5f 57 ed Y...J38...)W.
  0069 - d7 88 67 02 7a 31 4e a7-c0 c0 91 44 cf.de ...g.zi}).D.
  0078 - 6b ae 39 69 7e 0e 05-7e b0 f0 da d7 3b 8b k.9ip.....
  0087 - 3f 4e 42 6b 32 1b 45 38-7d 37 81 d9 82 a1 1a 0b 6.B.2.AS.....
  0096 - 39 43 ac 10 c9 7b 45 38-7d 37 81 d9 82 a1 1a 0b 49C...(T..8....
  00a5 - 56 10 e3 73 48 72 74 d6-9c 18 0a 4a 19 09 88 V.shrT....T...
  00b4 - 6c 64 f6 96 2d 42 32 4b-33 98 04 1a 4a 1e 1b 88 1d.-B.3....
  00c3 - dc 69 58 d5 3d 59 38 0b-b8 b5 2a a3 5e 22 99 88 .iX.=....."^\
  00d2 - 5d 0a ae 41 6f 86 f0 a8-8d 0e b7 a9 81 db 2b ]..AK.........
  00e1 - 83 e8 b9 4e 9d 50 bf 62-65 c2 34 8a 18 a1 69 ...N.P.be.4....i
  00f0 - aa cb 5a 37 a5 29 bd a2-f9 cb 10 ef dd 1f 14 ..27.)........
  00ff - 23 10 95 d8 79 64 63 7b-3d 3f b1 3c 68 4c cf #ydc(.?.<h..
  010e - f9 a1 90 69 69 69 ca 23-d1 d3 25 b7 a1 5c 58 ...i...%..\X
  011d - 29 f3 ea 03 8f 24 df 6b-23 18 03 77 d3 71 31 )}$.k#.w.q1
  012c - f7 5d b1 8f 41 f9 d8 5b-65 3d fa 46 fb 26 17 ]..A..[=.=.F.
  013b - 12 63 26 cc b8 83 34-57 75 23 52 c8 41 7a .c6*.4WUtR.Az
  014a - 09 44 84 d5 47 b6 b4 cb 51-2b 6a 9b eb 3a 0e 4d .D.KQ..j
  0159 - ba ed f1 bd 23 c6 90 c6-54 f7 eb 9c e9 85 2e ....#....T....
  0168 - 2f 5d 6d 06 48 ef 8b a3 3b-c6 c4 dd dc 7 afe4 /m........
  0177 - d3 57 47 37 d7 dc 4e 1e-93 77 0d 8f 4f 22 df .WG7....w.O".
  0186 - a6 1d 73 08 3c 32 c4 03-8c 1e b3 dd 33 83 a8 ...s.<2......3..
  0195 - 9a 87 95 e2 41 01 7c-b6 80 75 8c 04 10 69 ....A..[u..i
  01a4 - 48 98 60 fc 9f 49 0e 85-23 60 72 54 8b 32 49 H..I.#.rT2I
  01b3 - 69 8f 99 95 3a cf 1e c6-58 b7 a8 85 55 4a c4 ...X...T....
  01c2 - 49 70 1a 6d 4b 03 9e d1-03 dc 45 8f d4 b2 9c Ip.mK....E..
  01d1 - b0 4b 8c ed d5 40 c8 43-48 da 79 c1 86 56 65 .K..@.CH.y..V.
  01e0 - 18 8f 9f 3a 9e 4b 9b 84-0c 70 66 4b 90 29 6c ..K..pFK.J1
  01ef - 60 b7 ac 98 4e 91 8d 48-a0 9d bd df b2 81 fc ...N..H.......
  01fe - 86 25 10 db 59 df 9a 9d-c9 3f 10 f9 c6 d7 be %...Y...?..
authAttrs:
  <EMPTY>
mac:
  04dd - 3e a8 db >..
  0000 - f6 ff c6 ae f1 9c d2 -3d 98 5a 92 19 76 35 ........=.Z..v5
  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