Internet Engineering Task Force (IETF)
Request for Comments: 8591
Updates: 3261, 3428, 4975
Category: Standards Track
ISSN: 2070-1721

B. Campbell
Standard Velocity
R. Housley
Vigil Security

April 2019

SIP-Based Messaging with S/MIME

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 RFCs 3261, 3428, and 4975.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8591.

Table of Contents

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) system [CPM] uses the SIP MESSAGE method for short "pager mode" messages and uses MSRP for large messages and for sessions of messages. The Global System for Mobile Communications Association (GSMA) 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 the 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 [RFC8550] [RFC8551]. However, at the time of this writing, S/MIME is not in common use for SIP-based and MSRP-based messaging services. This document updates and clarifies RFCs 3261, 3428, and 4975 in an attempt to make S/MIME for SIP and MSRP easier to implement and deploy in an interoperable fashion.

This document updates RFCs 3261, 3428, and 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 embedded certificates in some situations. Finally, it updates RFCs 3261, 3428, and 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 applications. 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 a system would require considerable work around user enrollment, certificate and key generation and management, multi-party 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 [RFC8551] 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 Disposition Notification (IMDN) requests [RFC5438] 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 (UAs) SHOULD follow the conventions specified in [RFC8551] 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 [RFC8551] 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 }

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

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 [RFC8551] 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 [RFC8551] for the application/pkcs7-mime media type with smime-type=auth-enveloped-data.

Sending and receiving UAs MUST support the AES-128-GCM algorithm 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)
  csor(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)
  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 ECDH algorithm using curve25519 (X25519) [RFC7748] [RFC8418]. For convenience, (1) the identifier for the ECDH algorithm using the ANSI-X9.63-KDF with the SHA-256 algorithm and (2) the identifier 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 [RFC8550], 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 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 smime-type is 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 RFCs 3261 and 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 using the SIP MESSAGE method 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] requires that a SignedData message contain a certificate to be used to validate the signature. In order to reduce the message size, this document updates that text 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 (1) already has the certificate in its keychain or (2) has some other method of accessing the certificate.

7.2. SIP User Agent Capabilities

SIP UAs 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 (Unsupported Media Type) 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 User Agent Servers (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 UA 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 of [RFC3428] to add the statement that a UAC that receives a SIP MESSAGE request with an unsupported media type MUST return a 415 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 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 that 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 UA may break a message into chunks, which the receiving UA 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 UA to scan data to be sent in a specific chunk to ensure that the end-line does not accidentally occur as part of the 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 operations, so that each fragment appears as a distinct (separate) S/MIME message in MSRP. Such mechanisms are beyond the scope of 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 (MIME type not understood) 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 Section 7.3.1 of [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 UA 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 CPIM 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, IMDNs [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-based and RCS-based services include application servers that may need to insert timestamps 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 to validate signatures and decrypt enveloped data both (1) when those operations are applied to the entire CPIM body and (2) 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 Disposition Notifications

The 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 IMDNs. The IMDN mechanism 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, the IMDN mechanism 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 cleartext 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]
308202f606092a864886f70d010702a08202e7308202e3020101310d300b0609
6086480165030402013053060 92a864886f70d010701a0460444436f6e74656e
742d547970653a20746578742f706c61696e0d0a0d0a576174736f6e2c20636f
6d652068657265202d20492077616e7420746f2073656520796f752e0d0aa082
016b308201673082010da003020102020900b8793ec0e4c21530300a06082a86
48ce3d0403023026311430126035504 0a0c0b6578616d706c652e636f6d310e
300c06035504030c05416c696365301e170d313731323139323331323035 5a17
0d313831323139323331323035 5a3026311430126035504 0a0c0b6578616d70
6c652e636f6d310e300c06035504030c05416c6963653059301306072a8648ce
3d020106082a8648ce3d03010703420004d87b54729f2c22feebd9ddba0efa40
642297a6093887a4dae7990b23f87fa7ed99db8cf5a314f2ee64106ef1ed61db
fc0a4b91c953cbd022a751b914807bb794a32430223020603551d1104193017
86157369703a616c696365406578616d706c652e636f6d300a06082a8648ce3d
040302034800304502207879be1c27f846276fdf15e333e53c6f17a757388a02
cb7b8ae481c1641ae7a9022100ff99cd9c94076c82b02fea3b1350179a4b7752
e16fa30a3f9ab29650b0e2818931820109308201050201013033302631143012
060355040a0c0b6578616d706c652e636f6d310e300c06035504030c05416c69
6365020900b8793ec0e4c21530300b0609608648016503040201a06930180609
2a864886f70d010903310b06092a864886f70d010701301c06092a864886f70d
010905310f170d3139303132363036313335345a302f06092a864886f70d0109
0431220420ef778fc940d5e6dc2576f47a599b3126195a9f1a227adaf35fa22c
050d8d195a300a06082a8648ce3d040302044730450220 05fdc2b55b0f444a46
be468dfc7ef3b7de30019ef0952a223e8521890b35bb4e02210090e43a9d9846
cf2af8159c5c0ef48848fa2f39f998b1bb99b52a6fc6c776f2c8
[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.

```
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]
3082018706092a864886f70d010702a08201783082017402010131 0d300b0609
608648016503040201305306092a864886f70d010701a0460444436f6e74656e
742d547970653a20746578742f706c61696e0d0a0d0a576174736f6e2c20636f
6d652068657265202d20492077616e7420746f2073656520796f752e0d0a3182
0109308201050201013033302631143012060355040a0c0b6578616d706c652e
636f6d310e300c06035504030c05416c696365020900b8793ec0e4c21530300b
0609608648016503040201a069301806092a864886f70d010903310b06092a86
4886f70d010701301c06092a864886f70d010905310f170d3139303132363036
313335345a302f06092a864886f70d01090431220420ef778fc940d5e6dc2576
f47a599b3126195a9f1a227adaf35fa22c050d8d195a300a06082a8648ce3d04
03020447304502203607275592d30c8c5a931041a01804d60c638ac9a8080918
87172a0887c8d4aa022100cd9e14bd21817336e9052fe590af2e2bcde16dd3e9
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.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]
30820790060b2a864886f70d0109100117a082077f3082077b0201003182024f
3082024b020100303330263114301206035504 0a0c0b6578616d706c652e636f
6d310e300c06035504030c05416c69636502090083f50bb70bd5c40e300d0609
2a864886f70d0101010500048202007 59a61b4ddf1f1af24668005635e476110
fa2723c1b9e45484b6d33e8387de967dc5e0cafb35571a56a1975cb550e7be31
c131da80fb731024845babb8d64cac26040424d9330561c843999415dd644b3c
ad95072f71451393c99f282c4883bd0ccc5dd54b931464e00a6e55e592c51a68
de1062516ec7d3ca8e764bb8ac789a88377765ef8dc36c0a6ed3ecae5285cac6
a29d5059445719a1bdcf906e0ff37e2c2ef0f4ec6225100cc062e1c748963bbc
88b8e3dfcf714073729dd5c7583e758acf3d186f2fa417be22c37c9a76c6b427
29aad27f73ae44ac98474d1eeb48948c12a403d0b3ce08a218d6af456924897c
c5c9664f6dfeb3f18141158dfc3b84090aa60380aa865137e1699c5c81974167
9d7a3c90ba79e6d7d5c8d89bb54a667423e43b0b7d6f78c0b4ab67bc343662a6
35fe595f1149c53950cac2e0ba318c227e6f76a8d940400fd3d3ea1c8ecea003
dcce2f1fb00f5cea335de1303fcbf93d8e1cbfd682f19beb624bacd1d7b8f580
f114a13b890894fb4044a5daa764b7f8c5ff92949452b35aeb9639b8ad63c051
5c95ccc6f823c2201067ea2262413fef397d48f7b6143f842ae8e1a48cad3ae0
1abaa3cf9ee7e36620e05cca0611bfac00eef1a498f2d259b9f0f7da83ef6f1b
061f387c2dc48c8b5dbaca862308f32f47925165c9e5ebb467799884918dd697
b447f4c407989b889b0c2e9580af783082050f06092a864886f70d010701301e
06096086480165030401063011040c4d8757222eac5294117f0c120201108082
04e0fe2fb3de0bf06998c39bf4a952fabf8b0fee3d7e2e85181aecf1a89e1a2e
decd9404885612dfc6984334d8602b7749b2504e45f57c3b066626b0fc746236
1eec267c560139be5cd286a2af9696cf51852278e52c3818cab0a68c598de4fc
e14a333884e4de5ddf57edd78867027a31e4a7c0c0299144c5de6bae39699e70
0e057eb0f0dad73b8b369f42eb321b41538781d982a11a0b3943ac10c97b54ee
b73b38ec131afc5610e373487274d69cafa9541902886c64f6962d42eb33f904
1a4ae11b88dc6958d53df50b8bb52aa35e2299885d0aae416b86f0a88d0eb7a9
81dbb283e8b94e9d50bf6265c2348a18a169aacb5a37a529bda2f9cb10efddcf
14231095d87964637bd33fb13c68b4cff9a1906960c1ea2301d325b7a15c5829
f3ea038f24df6b23180377d37131f75db18f41f9d85b653dfa46bf2617126326
ccf1cb833457752352c8417a094484d7b64bcf51b26a9beb3a0ed4b9caf1bd23
c690c654f7eb9ce9852e2f6d068eef8ba33bc6c4dddca7aef4d3574737d7c4dc
```

```
   1e93770d8f4f22dea61d73083c32c4038c1eb3dd3383a89a8795e241c2ed7cb6
   80758c041069489860fc9f490e85236072548b3249698f99953acf1ec658b7aa
   85e554c449701a6d4b039ed103dc458df4b29cb04b8cedd540c84348da79c186
   56d5188f9f3a9e4b9b840c70664b90296c60b7ac984e918d48a09dbddfb281fc
   862510db59d9fa9dc93f10f9c6d7bef72931d184cad7ac13c1a5295fc89fe3bb
   7eb8e02085a828c5a138786e607ade4f5e8d4115909209ba878a79305a5316c2
   2229e42b886d06481c8473f9d51269e2af6341bce20f768e860d7784ed46150e
   04ff50cd209c5b127511369fe06bc4aa9a72d8f1fe4fcf0866d664b365ffa86e
   8c1b43e7a9212aecc16ca350a28efae25fac054dd934bfe7e5fa4f753aa41596
   8c7ebec439e0ac0270b4874a068d22484c09d9e8abe17f1372b4b2f65f1148e8
   933eda92e5d1774564963b391c3bbd9f1c27ffe36f832e05155fc39ee6652fa7
   b4188975ec5c67b32c9f213c8ac6b8e132a5a7c3bf74f016405cd8c201d10521
   93e186d44358de388d73211ba2f1792f3cfeb9bbde7211d26f56ab06e11ccc9c
   cde2b88cd8373773eafc37fd85b7a7a2bcaec752e617d6e01c02b86e9d9a40f3
   20462c5d66f8351716dcd6014bdf30a60f75fc0631c920845ed8c0bad35ddf19
   84f2241cd3b529dc1028845f8089543df4f1441ede36b1bf31af5afc8c2b708d
   50b645d4e7db88648c3eefe14765158fb0e8d3bb53ddcbe26d7124c6e1d992f8
   3230aa953376ee8c68109568e8571f0c9bbda48f4df306fe747f371175148f31
   832767cd766cf07b450cbf62cad2a7bd71f1f88233f116a1a7f3caf12f34bcf4
   0d21e79ffc9827221b68b080ff03ad782d6d6d07871676f798943e54f13fd75c
   89c0b4263bf10f56243f9e72ef3b3899a539d9a3ac5be2b69400a3cf8d196c5c
   ed697b2ed803b987a5ee85c5095b48da7a5b03b47e2b9fe4cd4bc3098e864e0c
   e7d467da99cd7f3a9e947b5eea77f7a6be16c8c7e9e0decc1ff132559c234321
   7b9c2950386e85d2942121086cdfa19658195be6d7f86bca9881b695082964f1
   2e7cf801025d6792c6882409414d703321ec83abd698d68956118713a0ff1272
   acbc9a6d148900c74c16921df9b38f29ec46d4f10060fffe5e36bbbacaf2d1ba
   d7dd057ed3e30ebcd69083f9d3a2a26ef90b751d6a1adfa0590db19da107cf3e
   a8db0410f6ffc6e1aef19cd23d985a921976352d
   [end-hex]
   -------dsdfoe38sd$
```

Figure 3: Signed and Encrypted Message in MSRP

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
   3082024b020100303330263114301206035504 0a0c0b6578616d706c652e636f
   6d310e300c06035504030c05416c69636502090083f50bb70bd5c40e300d0609
   2a864886f70d0101010500048202007 59a61b4ddf1f1af2466800563 5e476110
   fa2723c1b9e45484b6d33e8387de967dc5e0cafb35571a56a1975cb550e7be31
   c131da80fb731024845babb8d64cac26040424d9330561c843999415dd644b3c
   ad95072f71451393c99f282c4883bd0ccc5dd54b931464e00a6e55e592c51a68
   de1062516ec7d3ca8e764bb8ac789a88377765ef8dc36c0a6ed3ecae5285cac6
   a29d5059445719a1bdcf906e0ff37e2c2ef0f4ec6225100cc062e1c748963bbc
   88b8e3dfcf714073729dd5c7583e758acf3d186f2fa417be22c37c9a76c6b427
   29aad27f73ae44ac98474d1eeb48948c12a403d0b3ce08a218d6af456924897c
   c5c9664f6dfeb3f18141158dfc3b84090aa60380aa865137e1699c5c81974167
   9d7a3c90ba79e6d7d5c8d89bb54a667423e43b0b7d6f78c0b4ab67bc343662a6
   35fe595f1149c53950cac2e0ba318c227e6f76a8d940400fd3d3ea1c8ecea003
   dcce2f1fb00f5cea335de1303fcbf93d8e1cbfd682f19beb624bacd1d7b8f580
   f114a13b890894fb4044a5daa764b7f8c5ff92949452b35aeb9639b8ad63c051
   5c95ccc6f823c2201067ea2262413fef397d48f7b6143f842ae8e1a48cad3ae0
   1abaa3cf9ee7e36620e05cca0611bfac00eef1a498f2d259b9f0f7da83ef6f1b
   061f387c2dc48c8b5dbaca862308f32f47925165c9e5ebb467799884918dd697
   b447f4c407989b889b0c2e9580af783082050f06092a864886f70d010701301e
   0609608648016503040 1063011040c4d8757222eac5294117f0c120201108082
   04e0fe2fb3de0bf06998c39bf4a952fabf8b0fee3d7e2e85181aecf1a89e1a2e
   decd9404885612dfc6984334d8602b7749b2504e45f57c3b066626b0fc746236
   1eec267c560139be5cd286a2af9696cf51852278e52c3818cab0a68c598de4fc
   e14a333884e4de5ddf57edd78867027a31e4a7c0c0299144c5de6bae39699e70
   0e057eb0f0dad73b8b369f42eb321b41538781d982a11a0b3943ac10c97b54ee
   b73b38ec131afc5610e373487274d69cafa9541902886c64f6962d42eb33f904
   1a4ae11b88dc6958d53df50b8bb52aa35e2299885d0aae416b86f0a88d0eb7a9
   81dbb283e8b94e9d50bf6265c2348a18a169aacb5a37a529bda2f9cb10efddcf
   14231095d87964637bd33fb13c68b4cff9a1906960c1ea2301d325b7a15c5829
   [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]
f3ea038f24df6b23180377d37131f75db18f41f9d85b653dfa46bf2617126326
ccf1cb833457752352c8417a094484d7b64bcf51b26a9beb3a0ed4b9caf1bd23
c690c654f7eb9ce9852e2f6d068eef8ba33bc6c4dddca7aef4d3574737d7c4dc
1e93770d8f4f22dea61d73083c32c4038c1eb3dd3383a89a8795e241c2ed7cb6
80758c041069489860fc9f490e85236072548b3249698f99953acf1ec658b7aa
85e554c449701a6d4b039ed103dc458df4b29cb04b8cedd540c84348da79c186
56d5188f9f3a9e4b9b840c70664b90296c60b7ac984e918d48a09dbddfb281fc
862510db59d9fa9dc93f10f9c6d7bef72931d184cad7ac13c1a5295fc89fe3bb
7eb8e02085a828c5a138786e607ade4f5e8d4115909209ba878a79305a5316c2
2229e42b886d06481c8473f9d51269e2af6341bce20f768e860d7784ed46150e
04ff50cd209c5b127511369fe06bc4aa9a72d8f1fe4fcf0866d664b365ffa86e
8c1b43e7a9212aecc16ca350a28efae25fac054dd934bfe7e5fa4f753aa41596
8c7ebec439e0ac0270b4874a068d22484c09d9e8abe17f1372b4b2f65f1148e8
933eda92e5d1774564963b391c3bbd9f1c27ffe36f832e05155fc39ee6652fa7
b4188975ec5c67b32c9f213c8ac6b8e132a5a7c3bf74f016405cd8c201d10521
93e186d44358de388d73211ba2f1792f3cfeb9bbde7211d26f56ab06e11ccc9c
cde2b88cd8373773eafc37fd85b7a7a2bcaec752e617d6e01c02b86e9d9a40f3
20462c5d66f8351716dcd6014bdf30a60f75fc0631c920845ed8c0bad35ddf19
84f2241cd3b529dc1028845f8089543df4f1441ede36b1bf31af5afc8c2b708d
50b645d4e7db88648c3eefe14765158fb0e8d3bb53ddcbe26d7124c6e1d992f8
3230aa953376ee8c68109568e8571f0c9bbda48f4df306fe747f371175148f31
832767cd766cf07b450cbf62cad2a7bd71f1f88233f116a1a7f3caf12f34bcf4
0d21e79ffc9827221b68b080ff03ad782d6d6d07871676f798943e54f13fd75c
89c0b4263bf10f56243f9e72ef3b3899a539d9a3ac5be2b69400a3cf8d196c5c
ed697b2ed803b987a5ee85c5095b48da7a5b03b47e2b9fe4cd4bc3098e864e0c
e7d467da99cd7f3a9e947b5eea77f7a6be16c8c7e9e0decc1ff132559c234321
7b9c2950386e85d2942121086cdfa19658195be6d7f86bca9881b695082964f1
2e7cf801025d6792c6882409414d703321ec83abd698d68956118713a0ff1272
acbc9a6d148900c74c16921df9b38f29ec46d4f10060fffe5e36bbbacaf2d1ba
d7dd057ed3e30ebcd69083f9d3a2a26ef90b751d6a1adfa0590db19da107cf3e
a8db0410f6ffc6e1aef19cd23d985a921976352d
[end-hex]
-------op2nc9a$
```

Figure 4: Signed, Encrypted, and Chunked MSRP Message

11. IANA Considerations

This document has no IANA actions.

12. Security Considerations

The security considerations for S/MIME [RFC8550] [RFC8551] and elliptic curves in CMS [RFC5753] apply. The S/MIME-related security considerations for SIP [RFC3261], SIP MESSAGE [RFC3428], and MSRP [RFC4975] apply.

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

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 between this work and the techniques discussed in [RFC8224] and [RTP-Sec] 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 the 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 provides 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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3428] Campbell, B., Ed., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, DOI 10.17487/RFC3428, December 2002, <https://www.rfc-editor.org/info/rfc3428>.

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, DOI 10.17487/RFC3565, July 2003, <https://www.rfc-editor.org/info/rfc3565>.

[RFC3853] Peterson, J., "S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP)", RFC 3853, DOI 10.17487/RFC3853, July 2004, <https://www.rfc-editor.org/info/rfc3853>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4975] Campbell, B., Ed., Mahy, R., Ed., and C. Jennings, Ed., "The Message Session Relay Protocol (MSRP)", RFC 4975, DOI 10.17487/RFC4975, September 2007, <https://www.rfc-editor.org/info/rfc4975>.

[RFC5084] Housley, R., "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)", RFC 5084, DOI 10.17487/RFC5084, November 2007, <https://www.rfc-editor.org/info/rfc5084>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5753] Turner, S. and D. Brown, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 5753, DOI 10.17487/RFC5753, January 2010, <https://www.rfc-editor.org/info/rfc5753>.

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, DOI 10.17487/RFC5754, January 2010, <https://www.rfc-editor.org/info/rfc5754>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8418] Housley, R., "Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)", RFC 8418, DOI 10.17487/RFC8418, August 2018, <https://www.rfc-editor.org/info/rfc8418>.

[RFC8419] Housley, R., "Use of Edwards-Curve Digital Signature Algorithm (EdDSA) Signatures in the Cryptographic Message Syntax (CMS)", RFC 8419, DOI 10.17487/RFC8419, August 2018, <https://www.rfc-editor.org/info/rfc8419>.

[RFC8550] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Certificate Handling", RFC 8550, DOI 10.17487/RFC8550, April 2019, <https://www.rfc-editor.org/info/rfc8550>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[X680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1, August 2015, <https://www.itu.int/rec/T-REC-X.680>.

[X690] ITU-T, "Information Technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1, August 2015, <https://www.itu.int/rec/T-REC-X.690/>.

13.2. Informative References

[CPM] Open Mobile Alliance, "OMA Converged IP Messaging System Description, Candidate Version 2.2", September 2017.

[Efail] Poddebniak, D., Dresen, C., Muller, J., Ising, F., Schinzel, S., Friedberger, S., Somorovsky, J., and J. Schwenk, "Efail: Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels", August 2018, <https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-poddebniak.pdf>.

[RCS] GSMA, "RCS Universal Profile Service Definition Document, Version 2.2", May 2018, <https://www.gsma.com/futurenetworks/wp-content/uploads/2018/05/Universal-Profile-RCC.71-v2.2.pdf>.

[RFC3325] Jennings, C., Peterson, J., and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, DOI 10.17487/RFC3325, November 2002, <https://www.rfc-editor.org/info/rfc3325>.

[RFC3840] 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, <https://www.rfc-editor.org/info/rfc3840>.

[RFC3860] Peterson, J., "Common Profile for Instant Messaging (CPIM)", RFC 3860, DOI 10.17487/RFC3860, August 2004, <https://www.rfc-editor.org/info/rfc3860>.

[RFC3862] Klyne, G. and D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, DOI 10.17487/RFC3862, August 2004, <https://www.rfc-editor.org/info/rfc3862>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions for the Message Sessions Relay Protocol (MSRP)", RFC 4976, DOI 10.17487/RFC4976, September 2007, <https://www.rfc-editor.org/info/rfc4976>.

[RFC5438] Burger, E. and H. Khartabil, "Instant Message Disposition Notification (IMDN)", RFC 5438, DOI 10.17487/RFC5438, February 2009, <https://www.rfc-editor.org/info/rfc5438>.

[RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, DOI 10.17487/RFC6121, March 2011, <https://www.rfc-editor.org/info/rfc6121>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[RFC7701] Niemi, A., Garcia-Martin, M., and G. Sandbakken, "Multi-party Chat Using the Message Session Relay Protocol (MSRP)", RFC 7701, DOI 10.17487/RFC7701, December 2015, <https://www.rfc-editor.org/info/rfc7701>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[RTP-Sec] Peterson, J., Barnes, R., and R. Housley, "Best Practices for Securing RTP Media Signaled with SIP", Work in Progress, draft-ietf-sipbrandy-rtpsec-08, April 2019.

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   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: 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   .@d"...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
   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   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 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   l../.;.P..KwR.o
   003c - a3 0a 3f 9a b2 96 50 b0-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   .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 20 58 79 cc 62-85 e0 86 06 19 d3 bf   0E. Xy.b.......
   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.r..!...SD(M
   002d - 4c ef de 31 07 9c f9 71-bd 69 5d 6e c8 71 e9   L..1...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 embedded 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 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:
       <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 Embedded 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 Sections 10.3 and 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)
   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...N0U..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            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 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..O6
   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..|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.....$f..c^G
   000f - 61 10 fa 27 23 c1 b9 e4-54 84 b6 d3 3e 83 87   a..'#...T...>..
   001e - de 96 7d c5 e0 ca fb 35-57 1a 56 a1 97 5c b5   ..}....5W.V..\.
   002d - 50 e7 be 31 c1 31 da 80-fb 73 10 24 84 5b ab   P..1.1...s.$.[.
   003c - b8 d6 4c ac 26 04 04 24-d9 33 05 61 c8 43 99   ..L.&..$.3.a.C.
   004b - 94 15 dd 64 4b 3c ad 95-07 2f 71 45 13 93 c9   ...dK<.../qE...
   005a - 9f 28 2c 48 83 bd 0c cc-5d d5 4b 93 14 64 e0   .(,H....].K..d.
   0069 - 0a 6e 55 e5 92 c5 1a 68-de 10 62 51 6e c7 d3   .nU....h..bQn..
   0078 - ca 8e 76 4b b8 ac 78 9a-88 37 77 65 ef 8d c3   ..vK..x..7we...
   0087 - 6c 0a 6e d3 ec ae 52 85-ca c6 a2 9d 50 59 44   l.n...R.....PYD
   0096 - 57 19 a1 bd cf 90 6e 0f-f3 7e 2c 2e f0 f4 ec   W.....n..~,....
   00a5 - 62 25 10 0c c0 62 e1 c7-48 96 3b bc 88 b8 e3   b%...b..H.;....
   00b4 - df cf 71 40 73 72 9d d5-c7 58 3e 75 8a cf 3d   ..q@sr...X>u..=
   00c3 - 18 6f 2f a4 17 be 22 c3-7c 9a 76 c6 b4 27 29   .o/..."."|.v..')
   00d2 - aa d2 7f 73 ae 44 ac 98-47 4d 1e eb 48 94 8c   ...s.D..GM..H..
   00e1 - 12 a4 03 d0 b3 ce 08 a2-18 d6 af 45 69 24 89   ...........Ei$.
   00f0 - 7c c5 c9 66 4f 6d fe b3-f1 81 41 15 8d fc 3b   |..fOm....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 90 ba-79 e6 d7 d5 c8 d8 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 11 49 c5 39 50 ca   .46b.5.Y_.I.9P.
   013b - c2 e0 ba 31 8c 22 7e 6f-76 a8 d9 40 40 0f d3   ...1."~ov..@@..
   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 d7 b8 f5-80 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..9..c.Q\...
   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.
   01d1 - 06 1f 38 7c 2d c4 8c 8b-5d ba ca 86 23 08 f3   ..8|-...]...#..
   01e0 - 2f 47 92 51 65 c9 e5 eb-b4 67 79 98 84 91 8d   /G.Qe....gy....
   01ef - d6 97 b4 47 f4 c4 07 98-9b 88 9b 0c 2e 95 80   ...G...........
   01fe - af 78                                          .x
    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.....R..
   000f - 8b 0f 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-3b 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..&|V.9.\....
   004b - 96 96 cf 51 85 22 78 e5-2c 38 18 ca b0 a6 8c   ...Q."x.,8.....
   005a - 59 8d e4 fc e1 4a 33 38-84 e4 de 5d df 57 ed   Y....J38...].W.
   0069 - d7 88 67 02 7a 31 e4 a7-c0 c0 29 91 44 c5 de   ..g.z1....).D..
   0078 - 6b ae 39 69 9e 70 0e 05-7e b0 f0 da d7 3b 8b   k.9i.p..~......
   0087 - 36 9f 42 eb 32 1b 41 53-87 81 d9 82 a1 1a 0b   6.B.2.AS.......
   0096 - 39 43 ac 10 c9 7b 54 ee-b7 3b 38 ec 13 1a fc   9C...{T...8....
   00a5 - 56 10 e3 73 48 72 74 d6-9c af a9 54 19 02 88   V..sHrt....T...
   00b4 - 6c 64 f6 96 2d 42 eb 33-f9 04 1a 4a e1 1b 88   ld..-B.3...J...
   00c3 - dc 69 58 d5 3d f5 0b 8b-b5 2a a3 5e 22 99 88   .iX.=....*.^"..
   00d2 - 5d 0a ae 41 6b 86 f0 a8-8d 0e b7 a9 81 db b2   ]..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 cf 14   ..Z7.).........
   00ff - 23 10 95 d8 79 64 63 7b-d3 3f b1 3c 68 b4 cf   #...ydc{.?.<h..
   010e - f9 a1 90 69 60 c1 ea 23-01 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 bf 26 17   .]..A..[e=.F.&.
   013b - 12 63 26 cc f1 cb 83 34-57 75 23 52 c8 41 7a   .c&....4Wu#R.Az
   014a - 09 44 84 d7 b6 4b cf 51-b2 6a 9b eb 3a 0e d4   .D...K.Q.j..:..
   0159 - b9 ca f1 bd 23 c6 90 c6-54 f7 eb 9c e9 85 2e   ....#...T......
   0168 - 2f 6d 06 8e ef 8b a3 3b-c6 c4 dd dc a7 ae f4   /m.............
   0177 - d3 57 47 37 d7 c4 dc 1e-93 77 0d 8f 4f 22 de   .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 c2 ed 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..#.rT.2I
   01b3 - 69 8f 99 95 3a cf 1e c6-58 b7 aa 85 e5 54 c4   i...:...X....T.
   01c2 - 49 70 1a 6d 4b 03 9e d1-03 dc 45 8d f4 b2 9c   Ip.mK.....E....
   01d1 - b0 4b 8c ed d5 40 c8 43-48 da 79 c1 86 56 d5   .K...@.CH.y..V.
   01e0 - 18 8f 9f 3a 9e 4b 9b 84-0c 70 66 4b 90 29 6c   ...:.K...pfK.)l
   01ef - 60 b7 ac 98 4e 91 8d 48-a0 9d bd df b2 81 fc   ....N..H.......
   01fe - 86 25 10 db 59 d9 fa 9d-c9 3f 10 f9 c6 d7 be   .%..Y....?.....
   020d - f7 29 31 d1 84 ca d7 ac-13 c1 a5 29 5f c8 9f   .)1........)_..
   021c - e3 bb 7e b8 e0 20 85 a8-28 c5 a1 38 78 6e 60   ..~.....(..8xn.
   022b - 7a de 4f 5e 8d 41 15 90-92 09 ba 87 8a 79 30   z.O^.A.......y0
   023a - 5a 53 16 c2 22 29 e4 2b-88 6d 06 48 1c 84 73   ZS.").+.m.H..s
   0249 - f9 d5 12 69 e2 af 63 41-bc e2 0f 76 8e 86 0d   ...i..cA...v...
   0258 - 77 84 ed 46 15 0e 04 ff-50 cd 20 9c 5b 12 75   w..F....P...[.u
   0267 - 11 36 9f e0 6b c4 aa 9a-72 d8 f1 fe 4f cf 08   .6..k...r...O..
   0276 - 66 d6 64 b3 65 ff a8 6e-8c 1b 43 e7 a9 21 2a   f.d.e..n..C..!*
   0285 - ec c1 6c a3 50 a2 8e fa-e2 5f ac 05 4d d9 34   ..l.P...._..M.4
   0294 - bf e7 e5 fa 4f 75 3a a4-15 96 8c 7e be c4 39   ....Ou:....~..9
   02a3 - e0 ac 02 70 b4 87 4a 06-8d 22 48 4c 09 d9 e8   ...p..J.."HL...
   02b2 - ab e1 7f 13 72 b4 b2 f6-5f 11 48 e8 93 3e da   ....r..._.H..>.
   02c1 - 92 e5 d1 77 45 64 96 3b-39 1c 3b bd 9f 1c 27   ...wEd..9.....
   02d0 - ff e3 6f 83 2e 05 15 5f-c3 9e e6 65 2f a7 b4   ..o...._...e/..
   02df - 18 89 75 ec 5c 67 b3 2c-9f 21 3c 8a c6 b8 e1   ..u.\g.,.!<....
   02ee - 32 a5 a7 c3 bf 74 f0 16-40 5c d8 c2 01 d1 05   2....t..@\.....
```

```
   02fd - 21 93 e1 86 d4 43 58 de-38 8d 73 21 1b a2 f1   !....CX.8.s!...
   030c - 79 2f 3c fe b9 bb de 72-11 d2 6f 56 ab 06 e1   y/<....r..oV...
   031b - 1c cc 9c cd e2 b8 8c d8-37 37 73 ea fc 37 fd   ........77s..7.
   032a - 85 b7 a7 a2 bc ae c7 52-e6 17 d6 e0 1c 02 b8   .......R.......
   0339 - 6e 9d 9a 40 f3 20 46 2c-5d 66 f8 35 17 16 dc   n..@..F,]f.5...
   0348 - d6 01 4b df 30 a6 0f 75-fc 06 31 c9 20 84 5e   ..K.0..u..1...^
   0357 - d8 c0 ba d3 5d df 19 84-f2 24 1c d3 b5 29 dc   ....]....$...).
   0366 - 10 28 84 5f 80 89 54 3d-f4 f1 44 1e de 36 b1   .(._..T=..D..6.
   0375 - bf 31 af 5a fc 8c 2b 70-8d 50 b6 45 d4 e7 db   .1.Z..+p.P.E...
   0384 - 88 64 8c 3e ef e1 47 65-15 8f b0 e8 d3 bb 53   .d.>..Ge......S
   0393 - dd cb e2 6d 71 24 c6 e1-d9 92 f8 32 30 aa 95   ...mq$.....20..
   03a2 - 33 76 ee 8c 68 10 95 68-e8 57 1f 0c 9b bd a4   3v..h..h.W.....
   03b1 - 8f 4d f3 06 fe 74 7f 37-11 75 14 8f 31 83 27   .M...t.7.u..1..
   03c0 - 67 cd 76 6c f0 7b 45 0c-bf 62 ca d2 a7 bd 71   g.vl.{E..b....q
   03cf - f1 f8 82 33 f1 16 a1 a7-f3 ca f1 2f 34 bc f4   ...3......./4..
   03de - 0d 21 e7 9f fc 98 27 22-1b 68 b0 80 ff 03 ad   .!.....".h.....
   03ed - 78 2d 6d 6d 07 87 16 76-f7 98 94 3e 54 f1 3f   x-mm...v...>T.?
   03fc - d7 5c 89 c0 b4 26 3b f1-0f 56 24 3f 9e 72 ef   .\...&...V$?.r.
   040b - 3b 38 99 a5 39 d9 a3 ac-5b e2 b6 94 00 a3 cf   .8..9...[......
   041a - 8d 19 6c 5c ed 69 7b 2e-d8 03 b9 87 a5 ee 85   ..l\.i{........
   0429 - c5 09 5b 48 da 7a 5b 03-b4 7e 2b 9f e4 cd 4b   ..[H.z[..~+...K
   0438 - c3 09 8e 86 4e 0c e7 d4-67 da 99 cd 7f 3a 9e   ....N...g....:.
   0447 - 94 7b 5e ea 77 f7 a6 be-16 c8 c7 e9 e0 de cc   .{^.w..........
   0456 - 1f f1 32 55 9c 23 43 21-7b 9c 29 50 38 6e 85   ..2U.#C!{.)P8n.
   0465 - d2 94 21 21 08 6c df a1-96 58 19 5b e6 d7 f8   ..!!.l...X.[...
   0474 - 6b ca 98 81 b6 95 08 29-64 f1 2e 7c f8 01 02   k.......)d..|...
   0483 - 5d 67 92 c6 88 24 09 41-4d 70 33 21 ec 83 ab   ]g...$.AMp3!...
   0492 - d6 98 d6 89 56 11 87 13-a0 ff 12 72 ac bc 9a   ....V......r...
   04a1 - 6d 14 89 00 c7 4c 16 92-1d f9 b3 8f 29 ec 46   m....L......).F
   04b0 - d4 f1 00 60 ff fe 5e 36-bb ba ca f2 d1 ba d7   ......^6.......
   04bf - dd 05 7e d3 e3 0e bc d6-90 83 f9 d3 a2 a2 6e   ..~...........n
   04ce - f9 0b 75 1d 6a 1a df a0-59 0d b1 9d a1 07 cf   ..u.j...Y......
   04dd - 3e a8 db                                       >..
     authAttrs:
       <EMPTY>
     mac:
   0000 - f6 ff c6 e1 ae f1 9c d2-3d 98 5a 92 19 76 35   ........=.Z..v5
   000f - 2d                                             -
     unauthAttrs:
       <EMPTY>
```

Figure 8: Message Encrypted by Bob for Alice

Authors' Addresses

Ben Campbell
Standard Velocity, LLC

Email: ben@nostrum.com

Russ Housley
Vigil Security, LLC

Email: housley@vigilsec.com