Network Working Group
Internet-Draft
Updates: RFC 3261, RFC 3428, RFC 4975
(if approved)
Intended status: Standards Track
Expires: June 29, 2018

B. Campbell
Standard Velocity
R. Housley
Vigil Security

December 26, 2017

Securing Session Initiation Protocol (SIP) based Messaging with S/MIME
draft-campbell-sip-messaging-smime-02

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 June 29, 2018.

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) [CPM], [RCS] 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.

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 [RFC5750][RFC5751]. 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 the 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" send 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 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 version 3.2 [RFC5751] 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 is used to encrypt an encapsulated body part.

4.1. Signed Messages

While both SIP and MSRP require support for the multipart/signed format, this document recommends the use of application/pkcs7-mime 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 (UAs) SHOULD follow the conventions specified in [RFC5751] 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 [RFC5751] 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 P256 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][I-D.ietf-curdle-cms-eddsa-signatures]. For convenience, the ECDSA with SHA-256 algorithm identifier, the object identifier for the well-known NIST P256 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 P256 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  ::=  { 1 3 101 112 }
```

4.2. Encrypted Messages

When generating an encrypted message, sending UAs MUST follow the conventions specified in [RFC5751] for the application/pkcs7-mime media type with smime-type=enveloped-data. When decrypting a received message, receiving UAs MUST follow the conventions specified in [RFC5751] for the application/pkcs7-mime media type with smime-type=enveloped-data.

Sending and receiving UAs MUST support the AES-128-CBC for content encryption [RFC3565]. For convenience, the AES-128-CBC algorithm identifier is repeated here:

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

Sending and receiving UAs MAY support other content encryption algorithms.

Sending and receiving UAs MUST support the AES-128-WRAP 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 KEK identifier [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) using the NIST P256 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) using curve25519 (X25519) [RFC7748][I-D.ietf-curdle-cms-ecdh-new-curves]. For convenience, the ECDH using the ANSI-X9.63-KDF with SHA-256 algorithm identifier and the X25519 algorithm identifier are repeated here:

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

     id-X25519 OBJECT IDENTIFIER ::= { 1 3 101 110 }
```

4.3. Signed and Encrypted Messages

RFC 3261 section 23.2 says that when a 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 EnvelopedData object.

4.4. Certificate Handling

Sending and receiving UAs MUST follow the S/MIME certificate handling procedures [RFC5750], 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 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 P256 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".

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 received 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.

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. User Agent Capabilities

SIP user agents (UA) can 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.

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 "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 2XX 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. MRSP 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 answer. Thus, the 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 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, Intant 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 service 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.

If such clients need to provide 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. Some SIP header fields are folded to avoid over running the margins. Folded lines contain leading white space at the beginning of the line. These folds would not exist in the actual message.

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

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

Appendix A shows the detailed content of each S/MIME body.

10.1. Signed Message in SIP Including the Sender's Certificate

Figure 1 shows a message signed by Alice. This body uses the "application/pcks7-mime" media type with a 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 is 1300 octets. This is the maximum allowed for the SIP MESSAGE method unless the UAC has advance knowledge that no hop will use a transport protocol without congestion control.

```
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: 890

[start-hex]
3082037606092a864886f70d010702a0820367308203630201013l0d300b
0609608648016503040201305306092a864886f70d010701a0460444436f
6e74656e742d547970653a20746578742f706c61696e0d0a0d0a57617473
6f6e2c20636f6d652068657265202d20492077616e7420746f2073656520
796f752e0d0aa082016f3082016b30820110a003020102020090090238790
1727648e300a06082a8648ce3d040302302631143012060355040a0c0b65
78616d706c652e636f6d310e300c06035504030c05416c696365301e170d
313731323230323232353433395a170d313831323230323232353433395a3026
31143012060355040a0c0b6578616d706c652e636f6d310e300c06035504
030c05416c6963653059301306072a8648ce3d020106082a8648ce3d0301
0703420004d87b54729f2c22feebd9ddba0efa40642297a6093887a4dae7
990b23f87fa7ed99db8cf5a314f2ee64106ef1ed61dbfc0a4b91c953cbd0
22a751b914807bb794a327302530230603551d110101ff04193017861573
69703a616c696365406578616d706c652e636f6d300a06082a8648ce3d04
030203490030460221 00f16fe739ddf3a1ff072a78142395721f9c0629b5
8458644d855dad94da9b06f20221008ffda4ba4c65087584969bfb2002ba
f5eefebd693181b43666141f3639909884318201853082018102010130 33
3026311430120603550 40a0c0b6578616d706c652e636f6d310e300c0603
5504030c05416c6963650209009023879017276 48e300b06096086480165
03040201a081e4301806092a864886f70d010903310b06092a864886f70d
010701301c06092a864886f70d010905310f170d31373132323032323537
35315a302f06092a864886f70d01090431220420ef778fc940d5e6dc2576
f47a599b3126195a9f1a227adaf35fa22c050d8d195a307906092a864886
f70d01090f316c306a300b0609608648016503040 12a300b060960864801
6503040116300b0609608648016503040102300a06082a864886f70d0307
300e06082a864886f70d030202020080300d06082a864886f70d03020201
40300706052b0e030207300d06082a864886f70d0302020128300a06082a
8648ce3d040302044730450220 0f37c8d68628ed5a52e1208bb091999901
02f1de5766a45d5b4627fe4d87c9cc022100f0de29c03e7d3fcc5329b77f
e31faa10b0003c8249cb011cbb14410d4c9bf93e
[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 imbedded certificate. The resulting total length of 928 octets is 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: 518

[start-hex]
3082020206092a864886f70d010702a08201f3308201ef020101310d300b
0609608648016503040201305306092a864886f70d010701a0460444436f
6e74656e742d547970653a20746578742f706c61696e0d0a0d0a57617473
6f6e2c20636f6d652068657265202d20492077616e7420746f2073656520
796f752e0d0a3182018430820180020101303330263114301206035504 0a
0c0b6578616d706c652e636f6d310e300c06035504030c05416c69636502
0900b8793ec0e4c21530300b0609608648016503040201a081e430180609
2a864886f70d010903310b06092a864886f70d010701301c06092a864886
f70d010905310f170d3137313232313032313230345a302f06092a864886
f70d01090431220420ef778fc940d5e6dc2576f47a599b3126195a9f1a22
7adaf35fa22c050d8d195a307906092a864886f70d01090f316c306a300b
0609608648016503040 12a300b0609608648016503040116300b06096086
480165030401023 00a06082a864886f70d0307300e06082a864886f70d03
0202020080300d06082a864886f70d0302020140300706052b0e03020730
0d06082a864886f70d0302020128300a06082a8648ce3d04030204463044
022057773352edeed4ea693455e2a87b8b098decefe50ddb0ff7e391e84f
7976208a0220089cf365467a1a49e838b51f35a62c7a158e5fc999bf7d8f
bfb5262af5afec93
[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: 1-1567/1567
   Content-Disposition: attachment; filename="smime.p7m"
   Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
                 name="smime.p7m"

   [start-hex]
   3082061b06092a864886f70d010703a082060c308206080201003182024f
   3082024b020100303330263114301206035504 0a0c0b6578616d706c652e
   636f6d310e300c06035504030c05416c69636502090083f50bb70bd5c40e
   300d06092a864886f70d0101010500048202 00bbab785554a48e6248677b
   5c56328528282e172d36611dc2986ae168fc84d49f4120ea2cb895d5967f
   f35a22ed2e5fee4d7204e70c8697bf138d9fbd8485c300638f9ef93e6146
   5f1e1405fb5bf7b95f2faf12e441fb8cfcdfd5cf1d88d285cfa9fddf0de3
   f9c95b8cd750772924c7d919c80aa8677dc2bc63b5abe2a04e76ee0c2e6c
   041e08fa29476a4a76851944edd7fa79ada89709107bf65d56ac669b781a
   23f0fd7232de26bba07e1dca69f50118bd4955463d2cad403dc2a6749209
   dfc02c9e145270d5135ce5548bbf3347c6f356faa093feefdbba5d094f4a
   0e23a94686fca77cfa1759aaa4e27748227e6517063fbbd7a013abcb08f9
   50b2ac911b72b340d57c24d08e613f4e0a087821c820238e422e85bf3902
   c99a9629b0862945e00d1b433f6dc35e8d1cb5098597363624456dd867e6
   132d8ee935cc3b4124df6baba88770708af57c9ad70727410d6bf83ec0e5
   580e26c67f90608d375750ed93890256bf3f714fb676effcf363db0809b0
   21e90994a437353ee41432c7cf60e48cbd45420c659e75906cede2d44a5b
   b619014e73a0ba2d54ab3edbe23c63fad898411d1ac790552eadc66a358f
   bd4461efec935d0b8bbc2e6cf23e863a1ee7a4e7741f072c1d465ea1e6c3
   577b2e77acd1d1152f268235f85ae82f50871acf13e38b17fd69f88f973f
   6818682c4043b4ec7db17b22e20ee9becbf2c9f893308203ae06092a8648
   86f70d010701301d06096086480165030401020410 4d8757222eac529411
   7f0c12d671a127808203805d4077a1547c5f4699f07531a53eb88344d1a3
   b229ff91f3f69c0e94918c77c9f6bda194e35983ef9a38edca15678e65bd
   76ce665ca6e999b3a845e42666aa2703a5a4f0d3322d6de01e64545cbccf
   09e3c2268dfd86c336116b22cce80098619242fd482ece2fcd71a7c15ff7
   7bf133287df0ef7c51713bdc0f6cdada9198ea8fd81a9c5a50c5e9c0958b
   3347efc425038fb5b776ab469826227c697aecc6580d0a23a99e15b805ff
   3ba155c252f5e72bb9db133d049d992c18f4f4dad60a18dae729ba7c5836
   78ffb8604a8f7fe3cc62d0632cc66e1c4f9ba1fa9df56c6d9fd81f19c88b
   a6e9710bb0bbb0c5fcf9d2d5ea04d529fda78d60dc487d867c0e392174e9
   3ce2c3986cea7aef071e5b07b646c229f9069f27f3749456daea0a4e56bc
   491c9be370c544399a273d50c35f160fb37e5f7314c3d389a805c8e4776f
   0a2a89f555c9fe52080890abe2e39d2ed77a2b363d1c0fb375790028e962
   401230ae93aa4320a5da2ad7017c599508f79c3a0c35f9846e8a2c410a5e
   0d77e907c0151ce513e2e899de92ff8d177796238ade9309d75c976e9716
   ced4c45e1a339213d7b0824592394076f74a70454cc46565f4a836531646
   42827b2e28819ac3781afb529b7c72308b96978539d789d3d27cb1605b1a
   0ce966e9c6cb5825d235e523a6c2d948ef9314c902359adf03fe4684221f
```

```
afd1f833d759c6f2559b6e0a8897d64e42b49eae0e39dfaccc94ef3e733f
ca2212eb5ccbc7c5d7f042d02bf412d14c7ede0f664d799ea556f9763e74
2cacdd3efc8822bbe6c81fea27de6b0b06448252f9adeb6667b46056f39b
42f18f4d6258111fa243b5c39fdf8961bc6e59d8bd659d46f92a8ced04d1
a6af37e5c089b547a836df6994377cf92e8e74625569df6a6065f6c93bab
ef0d07cfac7af69d8bf87c96e6ebad2ebdb553f776e69143e706e227061e
5e3d0e38a83ab9c2ce62102f3021f7d8b9e56ad714514039f48d7fab85e
fbafee16e15d7afd8148ccfc9fb273f8bfd5bdebd0281a50095aa192c9e8
7a0f2c44bb57de5f86cfbda3337cd982dfae982a80879878646e03614515
cf94150479d20e3ce521617dda22d53a5829265564fa467e7db9e25f3d25
5a4f9f82fd9514ca177ac81b882acbe89d1cc640c7b980c5a9d5f70921bb
6fbf166d38aa04257e08c51b2df144e93363e0e47e8013df584b3f3130b4
df7c9ae17709f1bfd8ded1385741d80596b7b8d6a2f2e5a2f85029ce97ef
ed2c97f942f77b
[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: 1-780/1567
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
              name="smime.p7m"

[start-hex]
3082061b06092a864886f70d010703a082060c3082060802010003182024f
3082024b0201003033302631143012060355040a0c0b6578616d706c652e
636f6d310e300c06035504030c05416c69636502090083f50bb70bd5c40e
300d06092a864886f70d01010105000482020 0bbab785554a48e6248677b
5c56328528282e172d36611dc2986ae168fc84d49f4120ea2cb895d5967f
f35a22ed2e5fee4d7204e70c8697bf138d9fbd8485c300638f9ef93e6146
5f1e1405fb5bf7b95f2faf12e441fb8cfcdfd5cf1d88d285cfa9fddf0de3
f9c95b8cd750772924c7d919c80aa8677dc2bc63b5abe2a04e76ee0c2e6c
041e08fa29476a4a76851944edd7fa79ada89709107bf65d56ac669b781a
23f0fd7232de26bba07e1dca69f50118bd4955463d2cad403dc2a6749209
dfc02c9e145270d5135ce5548bbf3347c6f356faa093feefdbba5d094f4a
0e23a94686fca77cfa1759aaa4e27748227e6517063fbbd7a013abcb08f9
50b2ac911b72b340d57c24d08e613f4e0a087821c820238e422e85bf3902
```

```
c99a9629b0862945e00d1b433f6dc35e8d1cb5098597363624456dd867e6
132d8ee935cc3b4124df6baba88770708af57c9ad70727410d6bf83ec0e5
580e26c67f90608d375750ed93890256bf3f714fb676effcf363db0809b0
21e90994a437353ee41432c7cf60e48cbd45420c659e75906cede2d44a5b
b619014e73a0ba2d54ab3edbe23c63fad898411d1ac790552eadc66a358f
bd4461efec935d0b8bbc2e6cf23e863a1ee7a4e7741f072c1d465ea1e6c3
577b2e77acd1d1152f268235f85ae82f50871acf13e38b17fd69f88f973f
6818682c4043b4ec7db17b22e20ee9becbf2c9f893308203ae06092a8648
86f70d010701301d06096086480165030401020410 4d8757222eac529411
7f0c12d671a127808203805d4077a1547c5f4699f07531a53eb88344d1a3
b229ff91f3f69c0e94918c77c9f6bda194e35983ef9a38edca15678e65bd
76ce665ca6e999b3a845e42666aa2703a5a4f0d3322d6de01e64545cbccf
09e3c2268dfd86c336116b22cce80098619242fd482ece2fcd71a7c15ff7
[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: 781-1567/1567
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
              name="smime.p7m"

[start-hex]
7bf133287df0ef7c51713bdc0f6cdada9198ea8fd81a9c5a50c5e9c0958b
3347efc425038fb5b776ab469826227c697aecc6580d0a23a99e15b805ff
3ba155c252f5e72bb9db133d049d992c18f4f4dad60a18dae729ba7c5836
78ffb8604a8f7fe3cc62d0632cc66e1c4f9ba1fa9df56c6d9fd81f19c88b
a6e9710bb0bbb0c5fcf9d2d5ea04d529fda78d60dc487d867c0e392174e9
3ce2c3986cea7aef071e5b07b646c229f9069f27f3749456daea0a4e56bc
491c9be370c544399a273d50c35f160fb37e5f7314c3d389a805c8e4776f
0a2a89f555c9fe52080890abe2e39d2ed77a2b363d1c0fb375790028e962
401230ae93aa4320a5da2ad7017c599508f79c3a0c35f9846e8a2c410a5e
0d77e907c0151ce513e2e899de92ff8d177796238ade9309d75c976e9716
ced4c45e1a339213d7b0824592394076f74a70454cc46565f4a836531646
42827b2e28819ac3781afb529b7c72308b96978539d789d3d27cb1605b1a
0ce966e9c6cb5825d235e523a6c2d948ef9314c902359adf03fe4684221f
afd1f833d759c6f2559b6e0a8897d64e42b49eae0e39dfaccc94ef3e733f
ca2212eb5ccbc7c5d7f042d02bf412d14c7ede0f664d799ea556f9763e74
2cacdd3efc8822bbe6c81fea27de6b0b06448252f9adeb6667b46056f39b
42f18f4d6258111fa243b5c39fdf8961bc6e59d8bd659d46f92a8ced04d1
a6af37e5c089b547a836df6994377cf92e8e74625569df6a6065f6c93bab
ef0d07cfac7af69d8bf87c96e6ebad2ebdb553f776e69143e706e227061e
5e3d0e38a83ab9c2ce62102f3021f7d8b9e56ad6714514039f48d7fab85e
fbafee16e15d7afd8148ccfc9fb273f8bfd5bdebd0281a50095aa192c9e8
```

```
7a0f2c44bb57de5f86cfbda3337cd982dfae982a80879878646e03614515
cf94150479d20e3ce521617dda22d53a5829265564fa467e7db9e25f3d25
5a4f9f82fd9514ca177ac81b882acbe89d1cc640c7b980c5a9d5f70921bb
6fbf166d38aa04257e08c51b2df144e93363e0e47e8013df584b3f3130b4
df7c9ae17709f1bfd8ded1385741d80596b7b8d6a2f2e5a2f85029ce97ef
ed2c97f942f77b
[end-hex]
-------op2nc9a$
```

Figure 4: MSRP Chunked Signed and Encrypted Message

11. IANA Considerations

This document makes no requests of the IANA.

12. Security Considerations

The security considerations from S/MIME [RFC5750][RFC5751] and elliptic curves in CMS [RFC5753] apply. The S/MIME related security considerations from SIP [RFC3261][RFC3853], SIP MESSAGE [RFC3428], and MSRP [RFC4975] apply.

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 [RFC4474], [I-D.ietf-stir-rfc4474bis], 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 to white lists and blacklists.

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 vulnerabile to malicious or compromised intermediaries.

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

Certain messaging services, for example those based on CPM and RCS, may include intermediaries that attach metadata to user generated messages. In certain cases this metadata may reveal information to third parties that would have otherwise been encrypted. Implementors and operators should consider whether this metadata may create privacy leaks. Such an analysis is beyond the scope of this document.

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>.

[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>.

[RFC5750] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling", RFC 5750, DOI 10.17487/RFC5750, January 2010, <https://www.rfc-editor.org/info/rfc5750>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <https://www.rfc-editor.org/info/rfc5751>.

[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>.

[X680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, 2015.

[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, 2015.

13.2. Informative References

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

[I-D.ietf-curdle-cms-ecdh-new-curves] Housley, R., "Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)", draft-ietf-curdle-cms-ecdh-new-curves-10 (work in progress), August 2017.

[I-D.ietf-curdle-cms-eddsa-signatures] Housley, R., "Use of EdDSA Signatures in the Cryptographic Message Syntax (CMS)", draft-ietf-curdle-cms-eddsa-signatures-08 (work in progress), October 2017.

[I-D.ietf-sipbrandy-rtpsec] Peterson, J., Rescorla, E., Barnes, R., and R. Housley, "Best Practices for Securing RTP Media Signaled with SIP", draft-ietf-sipbrandy-rtpsec-03 (work in progress), October 2017.

[I-D.ietf-stir-rfc4474bis] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-stir-rfc4474bis-16 (work in progress), February 2017.

[RCS] GSMA, "RCS Universal Profile Service Definition Document, Version 2.0", June 2017.

[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>.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, DOI 10.17487/RFC4474, August 2006, <https://www.rfc-editor.org/info/rfc4474>.

[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>.

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: 10386294218579993742
          signature:
            algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
            parameter: <ABSENT>
          issuer: O=example.com, CN=Alice
```

```
        validity:
          notBefore: Dec 20 22:54:39 2017 GMT
          notAfter: Dec 20 22:54:39 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: TRUE
            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 46 02 21 00 f1 6f e7-39 dd f3 a1 ff 07 2a   0F.!..o.9.....*
  000f - 78 14 23 95 72 1f 9c 06-29 b5 84 58 64 4d 85   x.#.r...)..XdM.
  001e - 5d ad 94 da 9b 06 f2 02-21 00 8f fd a4 ba 4c   ].......!.....L
  002d - 65 08 75 84 96 9b fb 20-02 ba f5 ee fe bd 69   e.u.... ......i
  003c - 31 81 b4 36 66 14 1f 36-39 90 98 84            1..6f..69...
   crls:
     <EMPTY>
   signerInfos:
      version: 1
      d.issuerAndSerialNumber:
        issuer: O=example.com, CN=Alice
        serialNumber: 10386294218579993742
      digestAlgorithm:
        algorithm: sha256 (2.16.840.1.101.3.4.2.1)
        parameter: <ABSENT>
      signedAttrs:
          object: contentType (1.2.840.113549.1.9.3)
          value.set:
            OBJECT:pkcs7-data (1.2.840.113549.1.7.1)

          object: signingTime (1.2.840.113549.1.9.5)
          value.set:
```

```
            UTCTIME:Dec 20 22:57:51 2017 GMT

          object: messageDigest (1.2.840.113549.1.9.4)
          value.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

          object: S/MIME Capabilities (1.2.840.113549.1.9.15)
          value.set:
            SEQUENCE:
   0:d=0  hl=2 l= 106 cons: SEQUENCE
   2:d=1  hl=2 l=  11 cons:  SEQUENCE
   4:d=2  hl=2 l=   9 prim:   OBJECT            :aes-256-cbc
  15:d=1  hl=2 l=  11 cons:  SEQUENCE
  17:d=2  hl=2 l=   9 prim:   OBJECT            :aes-192-cbc
  28:d=1  hl=2 l=  11 cons:  SEQUENCE
  30:d=2  hl=2 l=   9 prim:   OBJECT            :aes-128-cbc
  41:d=1  hl=2 l=  10 cons:  SEQUENCE
  43:d=2  hl=2 l=   8 prim:   OBJECT            :des-ede3-cbc
  53:d=1  hl=2 l=  14 cons:  SEQUENCE
  55:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
  65:d=2  hl=2 l=   2 prim:   INTEGER           :80
  69:d=1  hl=2 l=  13 cons:  SEQUENCE
  71:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
  81:d=2  hl=2 l=   1 prim:   INTEGER           :40
  84:d=1  hl=2 l=   7 cons:  SEQUENCE
  86:d=2  hl=2 l=   5 prim:   OBJECT            :des-cbc
  93:d=1  hl=2 l=  13 cons:  SEQUENCE
  95:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
 105:d=2  hl=2 l=   1 prim:   INTEGER           :28
       signatureAlgorithm:
         algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
         parameter: <ABSENT>
       signature:
  0000 - 30 45 02 20 0f 37 c8 d6-86 28 ed 5a 52 e1 20   0E. .7...(.ZR.
  000f - 8b b0 91 99 99 01 02 f1-de 57 66 a4 5d 5b 46   .........Wf.][F
  001e - 27 fe 4d 87 c9 cc 02 21-00 f0 de 29 c0 3e 7d   '.M....!...).>}
  002d - 3f cc 53 29 b7 7f e3 1f-aa 10 b0 00 3c 82 49   ?.S).........<.I
  003c - cb 01 1c bb 14 41 0d 4c-9b f9 3e               .....A.L..>
       unsignedAttrs:
         <EMPTY>
```

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 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:
      <EMPTY>
    crls:
      <EMPTY>
    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)
            value.set:
              OBJECT:pkcs7-data (1.2.840.113549.1.7.1)

            object: signingTime (1.2.840.113549.1.9.5)
            value.set:
              UTCTIME:Dec 21 02:12:04 2017 GMT

            object: messageDigest (1.2.840.113549.1.9.4)
            value.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
```

```
            object: S/MIME Capabilities (1.2.840.113549.1.9.15)
            value.set:
              SEQUENCE:
     0:d=0  hl=2 l= 106 cons: SEQUENCE
     2:d=1  hl=2 l=  11 cons:  SEQUENCE
     4:d=2  hl=2 l=   9 prim:   OBJECT            :aes-256-cbc
    15:d=1  hl=2 l=  11 cons:  SEQUENCE
    17:d=2  hl=2 l=   9 prim:   OBJECT            :aes-192-cbc
    28:d=1  hl=2 l=  11 cons:  SEQUENCE
    30:d=2  hl=2 l=   9 prim:   OBJECT            :aes-128-cbc
    41:d=1  hl=2 l=  10 cons:  SEQUENCE
    43:d=2  hl=2 l=   8 prim:   OBJECT            :des-ede3-cbc
    53:d=1  hl=2 l=  14 cons:  SEQUENCE
    55:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
    65:d=2  hl=2 l=   2 prim:   INTEGER           :80
    69:d=1  hl=2 l=  13 cons:  SEQUENCE
    71:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
    81:d=2  hl=2 l=   1 prim:   INTEGER           :40
    84:d=1  hl=2 l=   7 cons:  SEQUENCE
    86:d=2  hl=2 l=   5 prim:   OBJECT            :des-cbc
    93:d=1  hl=2 l=  13 cons:  SEQUENCE
    95:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
   105:d=2  hl=2 l=   1 prim:   INTEGER           :28
          signatureAlgorithm:
            algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
            parameter: <ABSENT>
          signature:
   0000 - 30 44 02 20 57 77 33 52-ed ee d4 ea 69 34 55   0D. Ww3R....i4U
   000f - e2 a8 7b 8b 09 8d ec ef-e5 0d db 0f f7 e3 91   ..{............
   001e - e8 4f 79 76 20 8a 02 20-08 9c f3 65 46 7a 1a   .Oyv .. ...eFz.
   002d - 49 e8 38 b5 1f 35 a6 2c-7a 15 8e 5f c9 99 bf   I.8..5.,z.._...
   003c - 7d 8f bf b5 26 2a f5 af-ec 93                  }...&*....
          unsignedAttrs:
            <EMPTY>
```

Figure 6: Signed Message without Imbedded Certificate

A.3. Signed and Encrypted Message

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

A.3.1. Signed Message Prior to Encryption

```
 CMS_ContentInfo:
   contentType: pkcs7-signedData (1.2.840.113549.1.7.2)
   d.signedData:
```

```
    version: 1
    digestAlgorithms:
        algorithm: sha256 (2.16.840.1.101.3.4.2.1)
        parameter: <ABSENT>
    encapContentInfo:
      eContentType: pkcs7-data (1.2.840.113549.1.7.1)
      eContent:
  0000 - 43 6f 6e 74 65 6e 74 2d-54 79 70 65 3a 20 74   Content-Type: t
  000f - 65 78 74 2f 70 6c 61 69-6e 0d 0a 0d 0a 57 61   ext/plain....Wa
  001e - 74 73 6f 6e 2c 20 63 6f-6d 65 20 68 65 72 65   tson, come here
  002d - 20 2d 20 49 20 77 61 6e-74 20 74 6f 20 73 65    - I want to se
  003c - 65 20 79 6f 75 2e 0d 0a-                        e you...
    certificates:
      d.certificate:
        cert_info:
          version: 2
          serialNumber: 11914627415941064473
          signature:
            algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
            parameter: <ABSENT>
          issuer: O=example.org, CN=Bob
          validity:
            notBefore: Dec 20 23:07:49 2017 GMT
            notAfter: Dec 20 23:07:49 2018 GMT
          subject: O=example.org, CN=Bob
          key:
            algor:
              algorithm: id-ecPublicKey (1.2.840.10045.2.1)
              parameter: OBJECT:prime256v1 (1.2.840.10045.3.1.7)
            public_key:  (0 unused bits)
  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-                    ..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:
       <EMPTY>
     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)
             value.set:
               OBJECT:pkcs7-data (1.2.840.113549.1.7.1)

             object: signingTime (1.2.840.113549.1.9.5)
             value.set:
               UTCTIME:Dec 22 23:43:18 2017 GMT

             object: messageDigest (1.2.840.113549.1.9.4)
             value.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

             object: S/MIME Capabilities (1.2.840.113549.1.9.15)
             value.set:
               SEQUENCE:
    0:d=0  hl=2 l= 106 cons: SEQUENCE
    2:d=1  hl=2 l=  11 cons:  SEQUENCE
    4:d=2  hl=2 l=   9 prim:   OBJECT            :aes-256-cbc
   15:d=1  hl=2 l=  11 cons:  SEQUENCE
   17:d=2  hl=2 l=   9 prim:   OBJECT            :aes-192-cbc
   28:d=1  hl=2 l=  11 cons:  SEQUENCE
   30:d=2  hl=2 l=   9 prim:   OBJECT            :aes-128-cbc
   41:d=1  hl=2 l=  10 cons:  SEQUENCE
   43:d=2  hl=2 l=   8 prim:   OBJECT            :des-ede3-cbc
   53:d=1  hl=2 l=  14 cons:  SEQUENCE
   55:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
   65:d=2  hl=2 l=   2 prim:   INTEGER           :80
   69:d=1  hl=2 l=  13 cons:  SEQUENCE
   71:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
   81:d=2  hl=2 l=   1 prim:   INTEGER           :40
```

```
    84:d=1  hl=2 l=   7 cons:  SEQUENCE
    86:d=2  hl=2 l=   5 prim:   OBJECT            :des-cbc
    93:d=1  hl=2 l=  13 cons:  SEQUENCE
    95:d=2  hl=2 l=   8 prim:   OBJECT            :rc2-cbc
   105:d=2  hl=2 l=   1 prim:   INTEGER           :28
        signatureAlgorithm:
          algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
          parameter: <ABSENT>
        signature:
   0000 - 30 45 02 20 23 e1 e1 2f-c6 9c 7b c3 ae d0 67   0E. #../..{...g
   000f - 8a ab 25 71 16 dd 9a 82-7c 36 24 a2 fa e5 fa   ..%q....|6$....
   001e - 98 52 01 2b 98 c1 02 21-00 9b 8d 7c ad 9a f2   .R.+...!...|...
   002d - 09 e8 ac f7 00 aa a7 64-ef 32 d0 3a 47 16 42   .......d.2.:G.B
   003c - 79 04 54 90 53 e8 58 aa-6c 69 37               y.T.S.X.li7
        unsignedAttrs:
          <EMPTY>
```

Figure 7: Message Signed by Bob prior to Encryption

A.3.2. Encrypted Message

```
 CMS_ContentInfo:
   contentType: pkcs7-envelopedData (1.2.840.113549.1.7.3)
   d.envelopedData:
     version: <ABSENT>
     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 - bb ab 78 55 54 a4 8e 62-48 67 7b 5c 56 32 85   ..xUT..bHg{\V2.
   000f - 28 28 2e 17 2d 36 61 1d-c2 98 6a e1 68 fc 84   ((..-6a...j.h..
   001e - d4 9f 41 20 ea 2c b8 95-d5 96 7f f3 5a 22 ed   ..A .,......Z".
   002d - 2e 5f ee 4d 72 04 e7 0c-86 97 bf 13 8d 9f bd   ._.Mr.........
   003c - 84 85 c3 00 63 8f 9e f9-3e 61 46 5f 1e 14 05   ....c...>aF_...
   004b - fb 5b f7 b9 5f 2f af 12-e4 41 fb 8c fc df d5   .[.._/...A.....
   005a - cf 1d 88 d2 85 cf a9 fd-df 0d e3 f9 c9 5b 8c   .............[.
   0069 - d7 50 77 29 24 c7 d9 19-c8 0a a8 67 7d c2 bc   .Pw)$......g}..
   0078 - 63 b5 ab e2 a0 4e 76 ee-0c 2e 6c 04 1e 08 fa   c....Nv...l....
   0087 - 29 47 6a 4a 76 85 19 44-ed d7 fa 79 ad a8 97   )GjJv..D...y...
   0096 - 09 10 7b f6 5d 56 ac 66-9b 78 1a 23 f0 fd 72   ..{.]V.f.x.#..r
   00a5 - 32 de 26 bb a0 7e 1d ca-69 f5 01 18 bd 49 55   2.&..~..i....IU
```

```
   00b4 - 46 3d 2c ad 40 3d c2 a6-74 92 09 df c0 2c 9e   F=,.@=..t....,.
   00c3 - 14 52 70 d5 13 5c e5 54-8b bf 33 47 c6 f3 56   .Rp..\.T..3G..V
   00d2 - fa a0 93 fe ef db ba 5d-09 4f 4a 0e 23 a9 46   .......].OJ.#.F
   00e1 - 86 fc a7 7c fa 17 59 aa-a4 e2 77 48 22 7e 65   ...|..Y...wH"~e
   00f0 - 17 06 3f bb d7 a0 13 ab-cb 08 f9 50 b2 ac 91   ..?........P...
   00ff - 1b 72 b3 40 d5 7c 24 d0-8e 61 3f 4e 0a 08 78   .r.@.|$..a?N..x
   010e - 21 c8 20 23 8e 42 2e 85-bf 39 02 c9 9a 96 29   !. #.B...9....)
   011d - b0 86 29 45 e0 0d 1b 43-3f 6d c3 5e 8d 1c b5   ..)E...C?m.^...
   012c - 09 85 97 36 36 24 45 6d-d8 67 e6 13 2d 8e e9   ...66$Em.g..-..
   013b - 35 cc 3b 41 24 df 6b ab-a8 87 70 70 8a f5 7c   5.;A$.k...pp..|
   014a - 9a d7 07 27 41 0d 6b f8-3e c0 e5 58 0e 26 c6   ...'A.k.>..X.&.
   0159 - 7f 90 60 8d 37 57 50 ed-93 89 02 56 bf 3f 71   ..`.7WP....V.?q
   0168 - 4f b6 76 ef fc f3 63 db-08 09 b0 21 e9 09 94   O.v...c....!...
   0177 - a4 37 35 3e e4 14 32 c7-cf 60 e4 8c bd 45 42   .75>..2..`...EB
   0186 - 0c 65 9e 75 90 6c ed e2-d4 4a 5b b6 19 01 4e   .e.u.l...J[...N
   0195 - 73 a0 ba 2d 54 ab 3e db-e2 3c 63 fa d8 98 41   s..-T.>..<c...A
   01a4 - 1d 1a c7 90 55 2e ad c6-6a 35 8f bd 44 61 ef   ....U...j5..Da.
   01b3 - ec 93 5d 0b 8b bc 2e 6c-f2 3e 86 3a 1e e7 a4   ..]....l.>.:...
   01c2 - e7 74 1f 07 2c 1d 46 5e-a1 e6 c3 57 7b 2e 77   .t..,.F^...W{.w
   01d1 - ac d1 d1 15 2f 26 82 35-f8 5a e8 2f 50 87 1a   ..../&.5.Z./P..
   01e0 - cf 13 e3 8b 17 fd 69 f8-8f 97 3f 68 18 68 2c   ......i...?h.h,
   01ef - 40 43 b4 ec 7d b1 7b 22-e2 0e e9 be cb f2 c9   @C..}.{".......
   01fe - f8 93                                          ..
     encryptedContentInfo:
       contentType: pkcs7-data (1.2.840.113549.1.7.1)
       contentEncryptionAlgorithm:
         algorithm: aes-128-cbc (2.16.840.1.101.3.4.1.2)
         parameter: OCTET STRING:
   0000 - 4d 87 57 22 2e ac 52 94-11 7f 0c 12 d6 71 a1   M.W"..R......q.
   000f - 27                                             '
       encryptedContent:
   0000 - 5d 40 77 a1 54 7c 5f 46-99 f0 75 31 a5 3e b8   ]@w.T|_F..u1.>.
   000f - 83 44 d1 a3 b2 29 ff 91-f3 f6 9c 0e 94 91 8c   .D...).........
   001e - 77 c9 f6 bd a1 94 e3 59-83 ef 9a 38 ed ca 15   w......Y...8...
   002d - 67 8e 65 bd 76 ce 66 5c-a6 e9 99 b3 a8 45 e4   g.e.v.f\.....E.
   003c - 26 66 aa 27 03 a5 a4 f0-d3 32 2d 6d e0 1e 64   &f.'.....2-m..d
   004b - 54 5c bc cf 09 e3 c2 26-8d fd 86 c3 36 11 6b   T\.....&....6.k
   005a - 22 cc e8 00 98 61 92 42-fd 48 2e ce 2f cd 71   ".....a.B.H../.q
   0069 - a7 c1 5f f7 7b f1 33 28-7d f0 ef 7c 51 71 3b   .._.{.3(}..|Qq;
   0078 - dc 0f 6c da da 91 98 ea-8f d8 1a 9c 5a 50 c5   ..l.........ZP.
   0087 - e9 c0 95 8b 33 47 ef c4-25 03 8f b5 b7 76 ab   ....3G..%....v.
   0096 - 46 98 26 22 7c 69 7a ec-c6 58 0d 0a 23 a9 9e   F.&"|iz..X..#..
   00a5 - 15 b8 05 ff 3b a1 55 c2-52 f5 e7 2b b9 db 13   ....;.U.R..+...
   00b4 - 3d 04 9d 99 2c 18 f4 f4-da d6 0a 18 da e7 29   =...,.........)
   00c3 - ba 7c 58 36 78 ff b8 60-4a 8f 7f e3 cc 62 d0   .|X6x..`J....b.
   00d2 - 63 2c c6 6e 1c 4f 9b a1-fa 9d f5 6c 6d 9f d8   c,.n.O.....lm..
   00e1 - 1f 19 c8 8b a6 e9 71 0b-b0 bb b0 c5 fc f9 d2   ......q........
   00f0 - d5 ea 04 d5 29 fd a7 8d-60 dc 48 7d 86 7c 0e   ....)...`.H}.|.
```

```
   00ff - 39 21 74 e9 3c e2 c3 98-6c ea 7a ef 07 1e 5b   9!t.<...l.z...[
   010e - 07 b6 46 c2 29 f9 06 9f-27 f3 74 94 56 da ea   ..F.)...'.t.V..
   011d - 0a 4e 56 bc 49 1c 9b e3-70 c5 44 39 9a 27 3d   .NV.I...p.D9.'=
   012c - 50 c3 5f 16 0f b3 7e 5f-73 14 c3 d3 89 a8 05   P._...~_s......
   013b - c8 e4 77 6f 0a 2a 89 f5-55 c9 fe 52 08 08 90   ..wo.*..U..R...
   014a - ab e2 e3 9d 2e d7 7a 2b-36 3d 1c 0f b3 75 79   ......z+6=...uy
   0159 - 00 28 e9 62 40 12 30 ae-93 aa 43 20 a5 da 2a   .(.b@.0...C ..*
   0168 - d7 01 7c 59 95 08 f7 9c-3a 0c 35 f9 84 6e 8a   ..|Y....:.5..n.
   0177 - 2c 41 0a 5e 0d 77 e9 07-c0 15 1c e5 13 e2 e8   ,A.^.w.........
   0186 - 99 de 92 ff 8d 17 77 96-23 8a de 93 09 d7 5c   ......w.#.....\
   0195 - 97 6e 97 16 ce d4 c4 5e-1a 33 92 13 d7 b0 82   .n.....^.3.....
   01a4 - 45 92 39 40 76 f7 4a 70-45 4c c4 65 65 f4 a8   E.9@v.JpEL.ee..
   01b3 - 36 53 16 46 42 82 7b 2e-28 81 9a c3 78 1a fb   6S.FB.{.(...x..
   01c2 - 52 9b 7c 72 30 8b 96 97-85 39 d7 89 d3 d2 7c   R.|r0....9....|
   01d1 - b1 60 5b 1a 0c e9 66 e9-c6 cb 58 25 d2 35 e5   .`[...f...X%.5.
   01e0 - 23 a6 c2 d9 48 ef 93 14-c9 02 35 9a df 03 fe   #...H.....5....
   01ef - 46 84 22 1f af d1 f8 33-d7 59 c6 f2 55 9b 6e   F."....3.Y..U.n
   01fe - 0a 88 97 d6 4e 42 b4 9e-ae 0e 39 df ac cc 94   ....NB....9....
   020d - ef 3e 73 3f ca 22 12 eb-5c cb c7 c5 d7 f0 42   .>s?."..\.....B
   021c - d0 2b f4 12 d1 4c 7e de-0f 66 4d 79 9e a5 56   .+...L~..fMy..V
   022b - f9 76 3e 74 2c ac dd 3e-fc 88 22 bb e6 c8 1f   .v>t,..>.."....
   023a - ea 27 de 6b 0b 06 44 82-52 f9 ad eb 66 67 b4   .'.k..D.R...fg.
   0249 - 60 56 f3 9b 42 f1 8f 4d-62 58 11 1f a2 43 b5   `V..B..MbX...C.
   0258 - c3 9f df 89 61 bc 6e 59-d8 bd 65 9d 46 f9 2a   ....a.nY..e.F.*
   0267 - 8c ed 04 d1 a6 af 37 e5-c0 89 b5 47 a8 36 df   ......7....G.6.
   0276 - 69 94 37 7c f9 2e 8e 74-62 55 69 df 6a 60 65   i.7|...tbUi.j`e
   0285 - f6 c9 3b ab ef 0d 07 cf-ac 7a f6 9d 8b f8 7c   ..;......z....|
   0294 - 96 e6 eb ad 2e bd b5 53-f7 76 e6 91 43 e7 06   .......S.v..C..
   02a3 - e2 27 06 1e 5e 3d 0e 38-a8 3a b9 c2 ce 62 10   .'..^=.8.:...b.
   02b2 - 2f 30 21 f7 d8 b9 e5 6a-d6 71 45 14 03 9f 48   /0!....j.qE...H
   02c1 - d7 fa b8 5e fb af ee 16-e1 5d 7a fd 81 48 cc   ...^.....]z..H.
   02d0 - fc 9f b2 73 f8 bf d5 bd-eb d0 28 1a 50 09 5a   ...s......(.P.Z
   02df - a1 92 c9 e8 7a 0f 2c 44-bb 57 de 5f 86 cf bd   ....z.,D.W._...
   02ee - a3 33 7c d9 82 df ae 98-2a 80 87 98 78 64 6e   .3|.....*...xdn
   02fd - 03 61 45 15 cf 94 15 04-79 d2 0e 3c e5 21 61   .aE.....y..<.!a
   030c - 7d da 22 d5 3a 58 29 26-55 64 fa 46 7e 7d b9   }.".:X)&Ud.F~}.
   031b - e2 5f 3d 25 5a 4f 9f 82-fd 95 14 ca 17 7a c8   ._=%ZO.......z.
   032a - 1b 88 2a cb e8 9d 1c c6-40 c7 b9 80 c5 a9 d5   ..*.....@......
   0339 - f7 09 21 bb 6f bf 16 6d-38 aa 04 25 7e 08 c5   ..!.o..m8..%~..
   0348 - 1b 2d f1 44 e9 33 63 e0-e4 7e 80 13 df 58 4b   .-.D.3c..~...XK
   0357 - 3f 31 30 b4 df 7c 9a e1-77 09 f1 bf d8 de d1   ?10..|..w......
   0366 - 38 57 41 d8 05 96 b7 b8-d6 a2 f2 e5 a2 f8 50   8WA...........P
   0375 - 29 ce 97 ef ed 2c 97 f9-42 f7 7b               ).....,..B.{
     unprotectedAttrs:
       <EMPTY>
```

Figure 8

Authors' Addresses

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

Email: ben@nostrum.com

Russ Housley
Vigil Security
918 Spring Knoll Drive
Herndon, VA 20170
US

Email: housley@vigilsec.com