Abstract

Some services provided by intermediaries depend on their ability to inspect a message body in the Session Initiation Protocol (SIP). When sensitive information is included in the message body, a SIP User Agent (UA) needs to protect it from other intermediaries than those that the UA agreed to disclose it to. This document proposes a mechanism for securing information passed between an end user and intermediaries using S/MIME. It also proposes mechanisms for a UA to
discover intermediaries which need to inspect an S/MIME-secured message body, or to receive the message body with data integrity.

This specification is approved at the proposed standards level due to the IANA registration requirements. It is of sufficient quality for that level, however, the use of this mechanism in this specification are considered experimental.

Table of Contents

1. Introduction .................................................. 3
   1.1. Conventions used in this document .................... 3
2. Generating S/MIME-secured Message Body ...................... 3
   2.1. S/MIME-secured Message Body for Confidentiality ...... 3
   2.2. S/MIME-secured Message Body for Data Integrity ...... 5
3. Indicating the Target Proxy and Content .................... 5
   4.1. Discovery with Error Responses ....................... 7
5. Behavior of UAs and Proxy Servers .......................... 8
   5.1. UAC Behavior ..................................... 9
   5.2. UAS Behavior .................................... 11
   5.3. Proxy Behavior .................................. 12
6. Proxy-Required-Body Header .................................. 14
7. Message Examples ............................................. 15
   7.1. Message Examples of End-to-Middle Confidentiality ...... 15
   7.2. Message Examples of End-to-Middle Integrity .......... 19
8. Security Considerations ....................................... 21
   8.1. Impersonating a Proxy Server .......................... 21
   8.2. Tampering with a Message Body ........................ 21
   8.3. Tampering with the Label of the Target Content ...... 22
9. IANA Considerations ........................................... 22
   9.1. 'Proxy-Required-Body' Header ........................ 22
   9.2. '495 Signature Required’ Response Code ............. 22
   9.3. '496 Proxy Undecipherable’ Response Code ............. 23
   9.4. '380 Required to View Content-Type’ Warn-code ...... 23
10. Changes ....................................................... 23
11. Acknowledgments .............................................. 24
12. References .................................................... 24
   12.1. Normative References ................................ 24
   12.2. Informative References .............................. 25
Authors’ Addresses ............................................... 26
Intellectual Property and Copyright Statements ................ 27
1. Introduction

When a UA requires services provided by intermediaries that depend on the message body in request/response messages, end-to-end confidentiality currently has to be disabled. This problem is pointed out in Section 23 of [1]. Since such intermediaries are not always adjacent to the UA, this situation requires security between the UA and the intermediaries for the message body. We call this "end-to-middle security", where by "end" we mean a UA and by "middle" we mean an intermediary, typically a proxy server.

End-to-middle security, as well as end-to-end security, consists of peer authentication, data integrity, and data confidentiality. Peer authentication is required to achieve data integrity and data confidentiality respectively. The mechanisms of end-to-middle peer authentication are established with pre-existing mechanisms such as HTTP Digest authentication [8]. Therefore, this document focuses on mechanisms for providing data confidentiality and integrity for end-to-middle security to meet the requirements discussed in [2].

The proposed mechanisms are based on S/MIME [3], since the major requirement is to have little impact on standardized end-to-end security mechanisms defined in [1], the way of handling S/MIME-secure messages. The mechanisms consist of generating an S/MIME-secured message body and indicating the target message body is for a specific proxy server selected by the UA. In addition, this document describes a mechanism for a UA to discover the intermediary which needs to inspect an S/MIME-secured message body, or to receive the message body with data integrity.

1.1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [4].

2. Generating S/MIME-secured Message Body

2.1. S/MIME-secured Message Body for Confidentiality

For end-to-middle confidentiality, a UA MUST generate S/MIME CMS [5] EnvelopedData. Prior to generating it, a UA needs to identify the target proxy servers and obtain their credentials, such as their public key certificates or shared secrets. One method is shown in Section 4.

The structure of the S/MIME CMS EnvelopedData contains encrypted data
specified in the "encryptedContentInfo" field and its recipient list specified in the "recipientInfos" field. The encrypted data is encrypted with a content-encryption-key (CEK) and the recipient list contains the CEKs encrypted with different key-encryption-keys (KEKs), one for each recipient. The KEKs are either the public keys of each recipient or the shared keys between the UA and each recipient.

If the encrypted data is destined for one or more than one proxy server(s), the recipient list MUST contain only the proxy server(s). If the same encrypted data is shared with the user agent server (UAS) and proxy servers, the recipient list (the "recipientInfos" field) MUST be addressed to the UAS and the proxy servers (e.g., Proxy #1 and Proxy #2), as shown in Figure 1.

```
+-----------------------------------------------------------+
| The "recipientInfos" field                                |
+---------------------------------------------------------+|
| CEK encrypted with UAS’s KEK                            |
+---------------------------------------------------------+|
| CEK encrypted with Proxy #1’s KEK                       |
+---------------------------------------------------------+|
| CEK encrypted with Proxy #2’s KEK                       |
+---------------------------------------------------------+|
| The "encryptedContentInfo" field                         |
+---------------------------------------------------------+|
| Content encrypted with CEK to be shared with recipients  |
+---------------------------------------------------------+|
```

Figure 1: An Example Structure of EnvelopedData for Sharing

If there are multiple pieces encrypted data destined for each proxy server, the recipient list in each piece of encrypted data MUST contain the relevant proxy server. If a piece of encrypted data is destined for a proxy server and another piece of encrypted data for the UAS, the recipient of each piece of encrypted data MUST be each entity respectively, as shown in Figure 2. In order to concatenate more than one CMS EnvelopedData, the user agent client (UAC) MUST generate a "multipart/mixed" MIME body.
2.2. S/MIME-secured Message Body for Data Integrity

For end-to-middle data integrity, a UA SHOULD generate S/MIME CMS SignedData. A UA MAY generate a signature in the SIP Identity [9] header, only if the UA has its own public and private key. These mechanisms allow any entity to verify the data integrity, if it is able to access the UA’s public key. This is why the same mechanisms can be used in both end-to-middle and end-to-end data integrity.

Note: There are other mechanisms which could provide data integrity, such as S/MIME CMS AuthenticatedData, which requires that a UA obtains the credential of the recipient proxy server in advance. However, AuthenticatedData is not used in [1] and require a mechanism to securely transmit the credential from the proxy server to the UA. Thus, this document does not describe the use of S/MIME CMS AuthenticatedData.

3. Indicating the Target Proxy and Content

A UA needs a way to indicate the content which is expected to be viewed by a proxy server, in order for the proxy server to easily determine whether to process a MIME body and if so, which part. To meet this requirement, the UA SHOULD set a label to indicate the proxy server and its target content using a new SIP header, "Proxy-
Required-Body". This header consists of a proxy server’s hostname and one or more "content-id" parameter(s) pointing to the "Content-ID" MIME header [10] placed in the target body. If a UA needs to request multiple proxy servers to view the same message body, it SHOULD set multiple "Proxy-Required-Body" headers that contain the same "content-id" parameter. If a UA needs to request a proxy server to view multiple body parts that are nested, it MUST set the "content-id" parameter of the outer body first in the "Proxy-Required-Body" header.

This indication is not mandatory implementation, since the proxy server that has its own security policy attempts to view the message body due to their own services, regardless of the indication by UAs. Yet, this indication is useful for encrypted data to determine the target body that is decipherable only by the destined entity. On the other hand, the indication for signed data is usually useless because any entities can verify the signed data and the signed data is always protecting the whole message body. Therefore, a UA is NOT RECOMMENDED to set a indication using the "Proxy-Required-Body" header for signed data.

Note: There were three other options to label a body: a new SIP parameter to an existing SIP header, a new MIME header, or a new parameter to an existing MIME header.

1) Using a new parameter to Route header. Since a proxy server views this header when forwarding a request message, it seems to be a reasonable option. However, it cannot work with strict routing.

2) Using a new MIME header, "Content-Target", as proposed in a previous version of this draft. Since this option is not necessary as a generic mechanism of MIME, it is not preferred.

3) Using a new MIME parameter to "Content-Disposition". The same reason as above.

A UA has no way to get any specific acknowledgment of this indication. Even if a UA indicates a proxy server that is not along the signaling path, or that doesn’t support this mechanism, the UA doesn’t have any error response. The UA can only acknowledge the proxy server’s behavior or compliance through the service which requires proxy server’s inspection of the message body fails.

Note: Is "Proxy-Required-Body" an appropriate name? "Proxy-Allowed-Body" was suggested as the naming for this header. Since the intension of the header is to request, not just permit, the proxy server to view the message body when indicated, "Proxy-Required-Body" is to be more appropriate.
4. Discovering the Security Policies of Proxy Servers

A discovery mechanism for security policies of proxy servers is needed when a UA does not statically know which proxy servers or domains have such policies. Security policies require disclosure of data and/or verification in order to provide some services which needs UA’s compliance.

There are two ways in which a UA can learn the policies of the proxy servers. One is by receiving an error response from the proxy servers. The error response shows the violation of the policies, then a UAC can learn them. However, it is not applicable to the UAS because there is no way to react a response message. Alternatively, a policy server can provide a UAC and the UAS a package mentioning proxy’s policy as described in [11]. When a proxy server needs to inspect the message body contained in the response, it needs to learn the policies from a policy server before sending the response. This document covers only the former.

4.1. Discovery with Error Responses

When the proxy server receives a request that can not be accepted due to its condition, the proxy server MUST reject with an error response. If the request contains encrypted data and the proxy server cannot view the message body that has to be viewed in order to proceed, the proxy server MUST reject with a new error response, 496 (Proxy Undecipherable). The proxy’s public key certificate and Content-Type to be viewed SHOULD be contained with the error response. The proxy’s public key certificate SHOULD be set as an "application/pkix-cert" [6] MIME body. The Content-Type that the proxy server needs to view SHOULD be set in the Warning header with a new warn-code, 380.

When a UAC receives a 496 (Proxy Undecipherable) response, the UAC MUST check the respondent’s name in the public key certificate and the target Content-Type that the proxy server wants to view in the Warning header, if they exist.

Until the previous version, 493 (Undecipherable) error response had been proposed to be shared by the UAS and a proxy server. However, the reactions requesting the UAC are different, as pointed out in the SIP mailing list. On receiving the error response from the UAS, the UAC should totally renew "recipientInfos" by encrypted CEK with the KEK obtained from the error response. On the other hand, on receiving the error response from the proxy server, the UAC first should analyze the feature of the message body and the proxy-requiring Content-Type obtained from the Warning header. If the UAC decides to share the
message body with the UAS and the proxy server, the UAC will reuse the "recipientInfos" of the previous request and add encrypted CEK with the proxy’s KEK obtained from the error response to it. If the UAC decides to send two parts of the message body separately, the UAC will add the EnvelopedData that contains a message body for the proxy into the EnvelopedData in the previous request and construct a "multipart/mixed" MIME body.

If a digital signature is not attached to the message body in the request and the proxy server requires the integrity check, the proxy server MUST reject with a 495 (Signature Required) error response. This error response does not contain Content-Type that is required signature, since the attached signature to the whole body is always required. The proxy server MAY attach the signature to a "message/sipfrag" body, in order to set the name of the proxy server in the error response.

When a proxy server requires both disclosure and an integrity check of the message body in a request message and the message satisfies neither, the proxy server SHOULD send one error response at a time. When a proxy server cannot decrypt the message body in a request message and does not see if the signature is placed inside, a proxy server SHOULD send an error response only for requesting disclosure. After receiving a request message including encrypted data destined for the proxy server, it finds out whether the message has a signature attached and SHOULD send an error response for requesting signature when the message lacks it.

There are two ways to encrypt and sign data: encrypt data after signing, and encrypt data before signing. Although this document does not limit the way, it is more secure to encrypt data after signing. It is RECOMMENDED for a UA to recognize the 495 error response requiring the signature for the data prior to the encryption, if the encryption is needed.

This discovery mechanism requires two more message exchange for an error condition per each proxy server in the signaling path in order to establish a session between UAs. Since this causes a delay in session establishment, it is desirable that the UAs learn the security policies of the proxy servers in advance.

5. Behavior of UAs and Proxy Servers

We describe here the behavior of UAs and proxy servers in a model in which a proxy server that provides a firewall traversal service for voice and video, and a logging service for instant messages exists in a signaling path as shown in Figure 3. The instant messages assumes

```
+-----+     +-----+     +-----+     +-----+
|  C  |-----|  C  |-----| [C] |-----|  C  |
+-----+     +-----+     +-----+     +-----+
UA #1      Proxy #1    Proxy #2     UA #2
```

| C : Content that UA #1 allows the entities to inspect
| [C]: Content that UA #1 prevents the entity from inspecting

Figure 3: Deployment example

5.1. UAC Behavior

When a UAC (UA #1) sends an INVITE or a MESSAGE request including encrypted message body for end-to-middle confidentiality, it MUST generate S/MIME CMS EnvelopedData, and SHOULD specify the hostname of Proxy #1 and Content-ID of the S/MIME CMS EnvelopedData which is to be decrypted by Proxy #1 in the "Proxy-Required-Body" header.

If UA #1 decides to share the message body with the UAS (UA #2) and the proxy server (Proxy #1) that requires the inspection of the message body, UA #1 MUST list encrypted CEK with the Proxy #1’s KEK and encrypted CEK with the UA #2’s KEK at the "recipientInfos" of the CMS EnvelopedData. If UA #1 decides to set the message body separately, UA #1 MUST structure a "multipart/mixed" body that contains two CMS EnvelopedData: one encrypted for UA #2 and another encrypted for Proxy #1. UA #1 MUST set the value "optional" in the handling parameter of the "Content-Disposition" MIME header for the EnvelopedData destined for Proxy #1, in order to avoid unnecessary error conditions in UA #2. The "multipart/mixed" MIME body MUST have either the value "required" in the handling parameter or no handling parameter, since the default value is "required" as specified in [1].

This separate structure is useful when keying materials, such as keys used for Secure RTP (SRTP), are included in the SDP[14], UA #1 does not want to show the keying materials to Proxy #1, although Proxy #1 needs to view the SDP for the firewall traversal service.

If UA #1 sends an INVITE request including encrypted the SDP just for end-to-end, being unaware of the service provided by Proxy #1 that requires the inspection of the message body, UA #1 will get a 496 (Proxy Undecipherable) error response with the public key of Proxy #1 and the Warning header requiring the disclosure of "application/sdp". If UA #1 sends a MESSAGE request including encrypted content just for
end-to-end, being unaware of the Proxy #1’s service, UA #1 will get a 496 (Proxy Undecipherable) error response with the public key of Proxy #1 and no Warning header requiring Content-Type.

By obtaining the error response, UA #1 acknowledges that Proxy #1 requires the disclosure of a partial or the whole message body. If UA #1 decides to meet the requirement of Proxy #1, UA #1 generates CMS EnvelopedData and sets the "Proxy-Required-Body" header as described above. If the UA #1 decides to share the message body with the UA #2 and Proxy #1, UA #1 MUST update the "recipientInfos" of the previous request by adding encrypted CEK with Proxy #1’s KEK obtained from the error response. If UA #1 decides to set the message body separately for Proxy #1, UA #1 MUST structure a "multipart/mixed" body by adding the CMS EnvelopedData for Proxy #1.

When UA #1 sends a request message of which message body needs end-to-middle integrity, it SHOULD generate S/MIME CMS SignedData to attach a digital signature. UA #1 MAY specify the hostname of Proxy #1 and Content-ID of the CMS SignedData to be validated by Proxy #1 in the "Proxy-Required-Body" SIP header.

If UA #1 sends a MESSAGE request without the signature, being unaware of Proxy #1’s service that requires the verification of the message body, UA #1 will get a 495 (Signature Required) error response with no Warning header requiring Content-Type.

By obtaining the error response, UA #1 acknowledges that an entity in the signaling path, such as Proxy #1, requires the signature of the whole message body. If UA #1 decides to meet the requirement and has its own public key, UA #1 SHOULD generate the CMS SignedData to attach a signature by computing with its own private key.

When UA #1 sends a request and needs both end-to-middle confidentiality and integrity for the message body, it SHOULD first generate S/MIME CMS SignedData to attach the digital signature for the content, and then generate S/MIME EnvelopedData to encrypt the CMS SignedData. UA#1 SHOULD specify the hostname of Proxy#1 and the Content-ID of the CMS EnvelopedData destined for Proxy #1 in the "Proxy-Required-Body". UA#1 also MAY specify the Content-ID of the CMS SignedData following the Content-ID of the CMS EnvelopedData in the header.

Note: Encryption after signature is more secure than attaching a signature after encryption, generally because the signature outside is easily detachable.

If UA #1 needs the confidentiality of the SDP, and UA #1 knows that Proxy #1 needs to view the both SDPs in an INVITE request and a 200
OK response for the firewall traversal service, UA #1 MAY use the CEK reuse mechanism [15][16]. UA #1 indicates the identifier of the CEK to be reused at the "unprotectedAttrs" field of the CMS EnvelopedData in an INVITE request as showed in Figure 4.

```
+-------------------------------------------------------------+
| The "recipientInfos" field                                    |
| +-----------------------------------------------------------+ |
| | CEK encrypted with UA #1’s KEK                            |
| +-----------------------------------------------------------+ |
| | CEK encrypted with Proxy #1’s KEK                         |
| +-----------------------------------------------------------+ |
+-------------------------------------------------------------+

+-------------------------------------------------------------+
| The "encryptedContentInfo" field                             |
| +-----------------------------------------------------------+ |
| | Content encrypted with CEK #1 to be shared with recipients  |
| +-----------------------------------------------------------+ |
+-------------------------------------------------------------+

+-------------------------------------------------------------+
| The "unprotectedAttrs" field                                |
| +-----------------------------------------------------------+ |
| | Identifier of CEK #1                                      |
| +-----------------------------------------------------------+ |
+-------------------------------------------------------------+
```

Figure 4: EnvelopedData with CEK reuse in an INVITE request

5.2. UAS Behavior

When the UAS (UA #2) receives a request that contains a MIME body, UA #2 inspects the MIME body depending on the value of the handling parameter in "Content-Disposition" header. If the MIME body structures S/MIME, UA #2 first decrypts and/or validates it as usual. If the decryption and/or the validation is successful, UA #2 responds with a 200 OK. A 200 OK response is RECOMMENDED to have the same type of S/MIME CMS data. For example, if UA #2 receives an INVITE request with a MIME body that structures the CMS EnvelopedData to encrypt the SDP, it is RECOMMENDED to respond with a 200 OK with a MIME body that structures the CMS EnvelopedData to encrypt the SDP. If UA #2 receives an INVITE request with a MIME body that structures the CMS SignedData to attach the signature of the SDP, it is RECOMMENDED to respond with a 200 OK response with a MIME body that structures the CMS SignedData to attach the signature of the SDP. However, a 200 OK response to the MESSAGE request does not need to use the same type of S/MIME CMS data since the response does not contain any MIME body.

When the CMS EnvelopedData body of the request, destined for UA #2, contains the "unprotectedAttrs" attribute specifying the identifier of the CEK, UA #2 MAY acknowledge that UA #1 is requesting to reuse the CEK for the disclosure of the message body in the subsequent
requests or responses. By checking the "Proxy-Required-Body" header in the receiving request, UA #2 MAY know the destination (Proxy #1) and the Content-Type to be disclosed. If UA #2 accept the disclosure, it MAY keep the CEK with the identifier specified in the "unprotectedAttrs" attribute. If UA #2 receives an INVITE message specifying the CEK reuse, UA #2 MAY reuse the CEK (CEK #1) to encrypt a new CEK (CEK #2) for encrypting the SDP in a 200 OK response as showed in Figure 5

```
+-------------------------------------------------------------+
| The "recipientInfos" field                                  |
| +-----------------------------------------------------------+ |
| | CEK #2 encrypted with CEK #1                              |
| +-----------------------------------------------------------+ |
| The "encryptedContentInfo" field                            |
| +-----------------------------------------------------------+ |
| | Content encrypted with CEK #2 to be shared with recipients |
| +-----------------------------------------------------------+ |
```

Figure 5: EnvelopedData with CEK reuse in a 200 OK response

Even when UA #2 receives a request that does not use S/MIME, UA #2 sometimes needs end-to-middle confidentiality for the message body in a response, for example, the SDP offer/answer in a 200 response and ACK request. The behavior for generating S/MIME CMS data is the same as how UA #1 operates as described in Section 5.1, while the behavior for discovering the security policies of Proxy #1 can not be not supported.

5.3. Proxy Behavior

When the proxy server supporting this mechanism for its own security policies (Proxy #1) receives a message, it MUST inspect the "Proxy-Required-Body" header(s). If the header includes the Proxy #1’s hostname, Proxy #1 MUST inspect the body indicated by the "content-id" parameter. If multiple "content-id" parameters exist in the header, Proxy #1 MUST inspect the bodies in order. Even if the header does not include the Proxy #1’s hostname, nor the header exists, Proxy #1 MAY view the message body following its own security policies.

When the indicated body is CMS EnvelopedData, Proxy #1 MUST try to decrypt the "recipientInfos" field. If there is a piece of encrypted data for Proxy #1, Proxy #1 will succeed in obtaining the CEK to decrypt the encrypted content at the "encryptedContentInfo" field.

If Proxy #1 fails to decrypt the message body that is required to
view, it MUST respond with a 496 (Proxy Undecipherable) response, if it is a request, otherwise any existing dialog MUST be terminated. If Proxy #1 requires the disclosure of the SDP to view the port information for firewall traversal, the 496 response MUST include the Warning header, containing "Required to view 'application/sdp'". If Proxy #1 requires the disclosure of the whole message body for the message logging service, it MUST respond without the Warning header containing Content-Type.

For firewall traversal service, Proxy #1 does not care about the information only for UA #2, if UA #1 sets different port information for UA #2 and Proxy #1 separately on purpose or not. The firewall traversal service for UA #1 will fail. However, Proxy #1 care about the information even only for UA #2, if Proxy #1 provides a call admission control using codec information in SDP. Proxy #1 needs to view the SDP destined not only for itself, but also the SDP destined for UA #2 in order to confirm that both of the codec information are the same. In other words, Proxy #1 needs to police if UA #1 does not attempt to use a different codec that requires more bandwidth. After all, Proxy #1 will require disclosure of all the message body by setting no Warning header requiring Content-Type.

If Proxy #1 succeeds in this decryption, it MAY inspect the "unprotectedAttrs" field of the CMS EnvelopedData body. If the attribute gives the key's identifier, Proxy #1 MAY keep the CEK with its identifier until the lifetime of the CEK expires. If it receives subsequent messages within the lifetime, it MAY try to decrypt the type "KEKRecipientInfo" of the "RecipientInfo" attribute by using this CEK.

When the indicated content contains CMS SignedData body, Proxy #1 MUST validate the digital signature. If the verification fails, Proxy #1 SHOULD reject the subsequent procedure. It MAY respond with a 403 (Forbidden) response if the message is a request, otherwise any existing dialog MAY be terminated.

When Proxy #1 needs validate the data integrity of the content but the indicated body does not contain CMS SignedData body, Proxy #1 MUST respond with a 495 (Signature Required) response if the message is a request, otherwise any existing dialog MAY be terminated. A 495 response contain no Warning header requiring Content-Type to be attached a signature, since the signature of the whole body is always required, when the data integrity is required.

When Proxy #1 needs to validate the data integrity of the content and view it, but the indicated content is the CMS EnvelopedData, Proxy #1 does not see if the signature exists inside. First, Proxy #1 tries
to decrypt the CMS EnvelopedData. If the decryption fails, Proxy #1
MUST respond with 496 (Proxy Undecipherable) that contains its own
public key and no Warning header requiring a specific Content-Type.
After getting decipherable data, Proxy #1 inspects the content and
validate the signature, if it exists. If the signature for the whole
body does not exist, Proxy #1 MUST respond with 495 (Signature
Required) that contains no Warning header requiring a specific
Content-Type. If the encrypted data is attached with the signature
outside, Proxy #1 MAY first validate the signature, instead of
checking the existence of the signature inside.

When Proxy #1 forwards the request, it MAY delete the "Proxy-
Required-Body" header that contains its own hostname.

When a provider operating Proxy #1 does not allow any information
related to its security policies to be revealed to Proxy #2
serving UA #2, Proxy #1 MAY deletes the "Proxy-Required-Body"
header. However, when UA #1 sends the request to Proxy #1 via a
proxy server operated by another provider, there is no way to
conceal the header from the other proxy servers.

6. Proxy-Required-Body Header

The following syntax specification uses the augmented Backus-Naur
Form (BNF) as described in RFC-2234 [7]. The new header "Proxy-
Required-Body" is defined as a SIP header.

Proxy-Required-Body   = "Proxy-Required-Body" HCOLON required-proxy
SEMI target-body *(SEMI generic-param)
required-proxy        = host
target-body           = cid-param *(COMMA cid-param)
cid-param             = "cid" EQUAL content-id
content-id            = LDQUOT dot-atom "@" (dot-atom / host) RDQUOT
dot-atom              = atom *( "." atom )
atom                  = 1*( alphanumeric / "-" / "!" / "@" / "#" / "$" / "&" /
                        "*" / "" / "'" / """ / "^" / """ / "~" )

Information about the use of headers in relation to SIP methods and
proxy processing is summarized in Table 1.
### Table 1: Summary of header field use

The "where" column gives the request and response types in which the header field can be used. The values in the "where" column are as follows:

- **R**: The header field may appear in requests
- **100-699**: A numeral range indicates response codes with which the header field can be used.

The "proxy" column gives the operations a proxy may perform on the header field:

- **d**: A proxy can delete a header field value.
- **r**: A proxy must be able to read the header field, so it cannot be encrypted.

The next columns relate to the presence of a header field in a method:

- **o**: The header field is optional.
- **-**: The header field is not applicable.

#### 7. Message Examples

The following examples illustrate the use of the mechanism defined in the previous sections.

##### 7.1. Message Examples of End-to-Middle Confidentiality

In the following example, a UAC needs message content in a MESSAGE request to be confidential and it allows a proxy server to view the message body. Even though the Content-Length has no digit, the appropriate length is to be set. In the example message below, the text with the box of asterisks ("**") is encrypted:
MESSAGE alice@atlanta.example.com --> ssl.atlanta.example.com

MESSAGE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 MESSAGE
Date: Fri, 20 June 2003 13:02:03 GMT
Proxy-Required-Body: ssl.atlanta.example.com;
cid=1234@atlanta.example.com

Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
              name=smime.p7m
Content-Transfer-Encoding: binary
Content-ID: 1234@atlanta.example.com
Content-Disposition: attachment;filename=smime.p7m;handling=required
Content-Length: ...

******************************************************************
* (recipientInfos)                                              *
* RecipientInfo[0] for ssl.atlanta.example.com public key         *
* RecipientInfo[1] for Bob’s public key                          *
*                                                            *
* (encryptedContentInfo)                                         *
* Content-Type: text/plain                                       *
* Content-Length: ...                                           *
*                                                            *
* Hello.                                                        *
* This is confidential.                                         *
*                                                            *
******************************************************************

If the proxy server successfully views the message body, the UAC receives a 200 OK from the UAS normally. However, if a proxy server fails to view the message body, the UAC receives a 496 (Proxy Undecipherable) error response from the proxy server, as follows:
In the following example, a UA needs the SDP in an INVITE request to be confidential and it allows a proxy server to view the SDP.
INVITE alice@atlanta.example.com --> ssl.atlanta.example.com

INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Date: Fri, 20 June 2003 13:02:03 GMT
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Proxy-Required-Body: ssl.atlanta.example.com;
cid=1234@atlanta.example.com
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
    name=smime.p7m
Content-Transfer-Encoding: binary
Content-ID: 1234@atlanta.example.com
Content-Disposition: attachment;filename=smime.p7m;handling=required
Content-Length: ...

******************************************************************
* (recipientInfos)                                               *
* RecipientInfo[0] for ssl.atlanta.example.com public key        *
* RecipientInfo[1] for Bob’s public key                          *
*                                                                *
* (encryptedContentInfo)                                         *
* Content-Type: application/sdp                                  *
* Content-Length: 151                                            *
*                                                                *
* v=0                                                            *
* o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com*
* s=-                                                            *
* c=IN IP4 192.0.2.101                                          *
* t=0 0                                                          *
* m=audio 49172 RTP/AVP 0                                         *
* a=rtpmap:0 PCMU/8000                                          *
*                                                                *
******************************************************************

When the proxy server successfully views the SDP, and the UAS
responds with a 200 OK. The 200 OK is to be encrypted as follows:
200 OK alice@atlanta.example.com <-- ssl.atlanta.example.com

SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
To: Bob <sip:bob@biloxi.example.com>;tag=832123456
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
 name=smime.p7m
Content-Transfer-Encoding: binary
Content-ID: 1234@atlanta.example.com

******************************************************************
* (recipientInfos)                                               *
* RecipientInfo[0] for Alice’s public key                        *
*                                                             *
* (encryptedContentInfo)                                         *
* Content-Type: application/sdp                                  *
* Content-Length: 147                                           *
*                                                             *
* v=0                                                            *
* o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com*
* s=-                                                            *
* c=IN IP4 192.0.2.201                                           *
* t=0 0                                                          *
* m=audio 3456 RTP/AVP 0                                          *
* a=rtpmap:0 PCMU/8000                                          *
******************************************************************

7.2. Message Examples of End-to-Middle Integrity

In the following example, a UA needs the integrity of message content in a MESSAGE request to be validated by a proxy server before it views message content. Even though the Content-Length has no digit, the appropriate length is to be set.
If the proxy server successfully validates the integrity of the message body, the UAC normally receives a 200 OK from the UAS. However, if a proxy server does not receive a signature for the whole message body, the UAC receives a 495 (Signature Required) error response from the proxy server, as follows:
8. Security Considerations

8.1. Impersonating a Proxy Server

The discovery mechanism in Section 4 relies on error responses, such as 495 (Signature Required) and 496 (Proxy Undecipherable). As for the 495 response, the responder is not critical from the security perspective, since it does not require any kind of downgrading security, but upgrading security by attaching the signature that can be validated by any entities. On the other hand, the 496 response is critical and vulnerable to be forged by a malicious user, since it is attached with the public key certificate that requires the disclosure of the whole or the partial message body to the UA.

To make sure that the 496 response is sent by a proper proxy server, a UA MUST authenticate the responder. Although the UA does not know what name is a proper proxy server, the UA MUST check if the common name of the public key certificate attached with the response corresponds to the proxy’s name in the domain where the UA connects to, or the domain where the recipients connect to. Additionally, a UA MUST verify the identifier of the proxy server and chains to a trusted certificate authority of the public key certificate. If a UA fails to check the correspondence and/or the verification, the public key certificate is presumably replaced or forged by a malicious user.

8.2. Tampering with a Message Body

This document describes a mechanism to encrypt data for multiple recipients, such as multiple proxy servers, or a recipient UA and proxy servers. A piece of encrypted data is decipherable and vulnerable to tampering by proxy servers at the previous hops.

In order to prevent such tampering, the UA SHOULD protect the data
integrity before encryption, when the encrypted data is meant to be shared with multiple proxy servers, or to be shared with the UAS and selected proxy servers. The UA SHOULD generate S/MIME CMS SignedData and then SHOULD generate the EnvelopedData to encrypt attached data with a digital signature. The recipient entity SHOULD verify the signature to see if the encrypted data has been modified after decryption by an entity listed in the "recipientInfos" field.

8.3. Tampering with the Label of the Target Content

This document also describes a new SIP header for labeling a message body for a proxy server. If a malicious user or proxy server modified/added/deleted the label, the specified message body is not inspected by the specified proxy server, and some services requiring its content cannot be provided. Or a proxy server will conduct an unnecessary processing on message bodies such as unpacking MIME structure, and/or signature verification. This is a possible cause for a Denial-of-Services attack to a proxy server.

To prevent such attacks, data integrity for the label is needed. UAs and proxy servers SHOULD use TLS mechanism to communicate with each other. Since a proxy server trusted to provide SIP routing is basically trusted to process SIP headers other than those related to routing, hop-by-hop security is reasonable to protect the label. In order to further protect the integrity of the label, UAs MAY generate a "message/sipfrag" body and attach a digital signature for the whole body.

9. IANA Considerations

This document requests requests to register a SIP header, two SIP response codes, and a SIP warn-code in the SIP parameters IANA registry.

9.1. ‘Proxy-Required-Body’ Header

This section includes the registration information for the ‘Proxy-Required-Body’ header which is described in Section 6 of this document.
Header Name: Proxy-Required-Body
Compact Form: (none)

9.2. ‘495 Signature Required’ Response Code

This section includes the registration information for the ‘495 Signature Required’ response code which is described in Section 4 of this document.
Response Code Number: 495
Default Response Phrase: Signature Required

9.3. ’496 Proxy Undecipherable’ Response Code

This section includes the registration information for the ’496 Proxy Undecipherable’ response code which is described in Section 4 of this document.
Response Code Number: 496
Default Response Phrase: Proxy Undecipherable Code

9.4. ’380 Required to View Content-Type’ Warn-code

This section includes the registration information for the ’380 Required to view Content-Type’ warn-code which is described in Section 4 of this document.
Warning Code Number: 380
Default Warning Phrase: Required to View Content-Type

10. Changes

Changes from -02.
  o Added text in the abstract.
  o Fixed the order of CMS data fields in the examples.
  o Modified the default response phrase for the 496 response code for the consistency with that of the 493 response code.
  o Added generic-params to the ”Proxy-Required-Body” header for supporting extension parameters.
  o Added REFER and PUBLISH methods in the table.
  o Clarified a new parameter and responses in the section of IANA consideration.

Changes from -01.
  o Changed an author’s contact address.

Changes from -00.
  o Added several figures that show the abstract of the structure of EnvelopedData.
  o Changed a error response that Proxy sends back in decryption failure from 493 (Undecipherable) to 496 (Proxy Indecipherable), a new one.
  o Changed the constraint of indicating CMS SignedData for a UA, from SHOULD to MAY/NOT RECOMMENDED.
  o Added the way that Proxy requires the disclosure for the whole body.
o Added the way that Proxy sets its own name to a 495 response.
o Corrected the applicability of the "Proxy-Required-Body" for ACK and PRACK.
o Removed the parameters for the CEK reuse from the message examples.
o Added text for detecting forged error response at impersonating proxy server in Security Consideration.

11. Acknowledgments

Thanks to Rohan Mahy and Cullen Jennings for their initial support of this concept and to many people for useful comments, especially Jon Peterson, Jonathan Rosenberg, Eric Burger, Russ Housely, Marjou Xavier, and Eric Rescorla.

12. References

12.1. Normative References


12.2. Informative References


Authors’ Addresses

Kumiko Ono
Columbia University
Department of Computer Science
New York, NY 10027
USA

Email: kumiko@cs.columbia.edu

Shinya Tachimoto
Network Service Systems Laboratories, NTT Corporation
Musashino-shi, Tokyo 180-8585
Japan

Email: tachimoto.shinya@lab.ntt.co.jp
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.