SIP
Internet-Draft
Expires: January 8, 2008

K. Ono
Columbia University
S. Tachimoto
NTT Corporation

July 7, 2007

End-to-middle Security in the Session Initiation Protocol (SIP)
draft-ietf-sip-e2m-sec-06

Status of this Memo

Copyright Notice

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. Is is of sufficient quality for that level, however, the use of this mechanism in this specification are considered experimental.

Table of Contents

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 [9]. 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 which contains encrypted data. Prior to generating it, a UA needs to identify contents to be encrypted, 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.

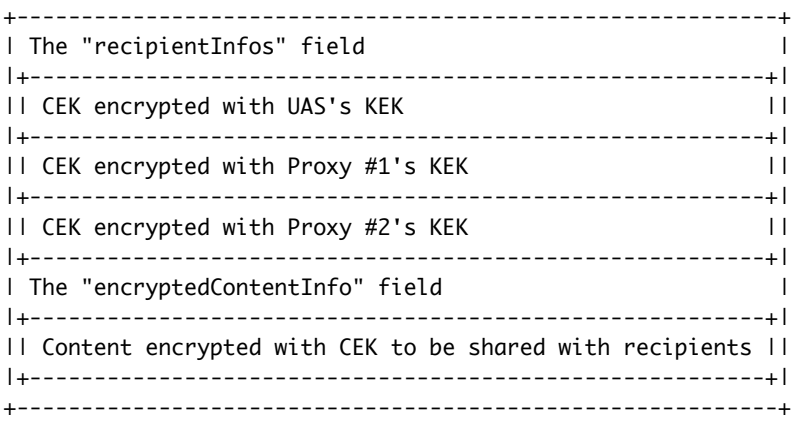


Figure 1: An Example Structure of EnvelopedData for Sharing

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.

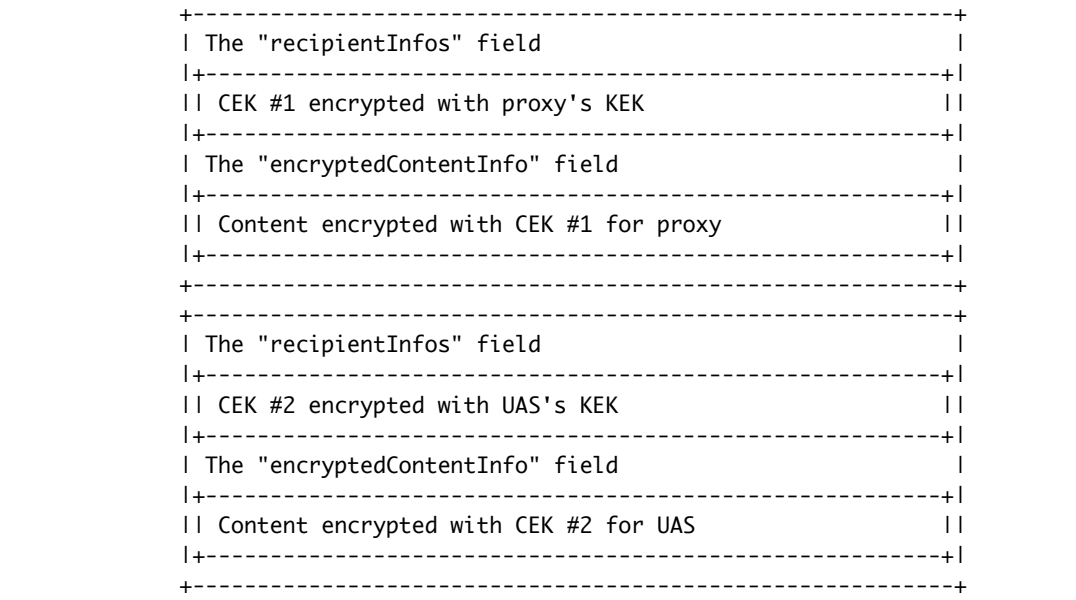


Figure 2: An Example Structure of EnvelopedData not for Sharing

2.2. S/MIME-secured Message Body for Data Integrity

For end-to-middle data integrity, a UA SHOULD generate S/MIME CMS SignedData to attach a digital signature for the target data. If this is not done, then as an alternative, a UA MAY generate the signature in the SIP Identity [10] mechanism. Generating the signature requires the generator, i.e., the UA, has its own public and private key pair that the UA is not required to have. 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, or CMS DigestedData as defined in [5]. However, S/MIME CMS AuthenticatedData requires that a UA obtains the credential of the recipient proxy server in advance. Thus, it requires a mechanism to securely transmit the credential from the proxy server to the UA. On the other hand, S/MIME CMS DigestedData does not require such mechanism. However, anybody can generate the digest data for a message. Then, it does not offer the data integrity from the originator. Additionally, neither of them is used in [1]. Therefore, a UA MUST NOT use either of S/MIME CMS AuthenticatedData or CMS DigestedData for interoperability.

2.3. S/MIME-secured Message Body for Confidentiality and Data Integrity

Both for end-to-middle confidentiality and the data integrity, a UA SHOULD first generate the signature as CMS SignedData, and encrypt the message containing the signature inside. Alternatively, a UA MAY generate the signature in the SIP identity which is set outside. As the third alternative, a UA MAY first encrypt the message, and attach the signature as CMS SignedData as defined in [1].

> There are two ways to encrypt and sign data: encrypt data after signing, and encrypt data before signing. It is more secure to encrypt data after signing than attaching a signature after encryption, generally because the signature outside is easily detachable. If the proxy server does not accept any message without the SIP identity, encrypting data before signing is secure enough.

3. Indicating the Target Proxy and Content

A UA need 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 MUST set a label to indicate the proxy server and its target content using a new SIP header, "Proxy-Inspect-Body" for encrypted data. The UA MAY omit to set the "Proxy-Inspect-Body" header for signed data. This header consists of a proxy server's hostname and one or more "cid" parameter(s) pointing to the "Content-ID" MIME header [11] placed in the target body.

This indication is useful for encrypted data, but less for signed data. The hostname field for encrypted data is useful for a proxy server to determine if the proxy server is destined to view the body. The "cid" parameter(s) for encrypted data is useful for a proxy server to specify the target part of a "multi-part/mixed" body to decrypt. On the contrary, the "cid" parameter for signed data always indicates the whole message body, since the signed data is always protecting the whole body. Additionally, since any entities can verify the signed data, the hostname indication for singed data is also less useful than the indication for encrypted data.

If a UA needs to request multiple proxy servers to view the same message body, it MUST set multiple "Proxy-Inspect-Body" headers that contain the same "cid" parameter. If a UA needs to request a proxy server to view multiple body parts that are nested, it MUST set the "cid" parameter of the outer body first and that of the inner body next in the "Proxy-Inspect-Body" header.

Note: The "cid" parameter indicates which part of the body is expected to be viewed by a proxy server, not what kind of content. If the message body is encrypted, the proxy server needs to decrypt it to see if the message body contains a necessary Content-Type, such as application/sdp.

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.

When a proxy server receives a message with the "Proxy-Inspect-Body" containing its hostname, the proxy server MUST try decrypting the content if the content is CMS EnvelopedData, or MUST validate the signature if the content is CMS SignedData. If it fails to decrypt or to validate, it MUST respond with an error code as described in Section 5.3.

When a proxy server receives a message missing the "Proxy-Inspect-Body" header containing its hostname, the proxy server misses viewing the message body. If the proxy server needs to avoid such failure for their own services, the proxy server MAY attempt to view the message body, regardless of the hostname field. No error code is defined to inform the UA that the indication was missing.

A UA has no way to get any specific acknowledgment of this indication. If a UA indicates a proxy server that is not along the signaling path, or that does not support this mechanism, the UA does not 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.

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.

A proxy security policy includes specification of the MIME types of message bodies that the proxy inspects as part of providing service. For each of these MIME types, if the proxy is unable to examine the corresponding message body, the proxy's security policy MAY require the proxy to reject the message with an error. Any message body for which an error results when the proxy cannot view the message body is called a proxy required message body (PRMB). The goal of security policy discovery is for the UA to learn the set of proxy required message bodies (PRMBs) for each proxy involved in the call, so that it can make those PRMBs visible to the proxies that require them. Configuration of proxy security policy, including selection of the MIME types that correspond to PRMBs is the responsibility of the proxy administrator.

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 [12]. 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 does not satisfy the proxy's security policy, the proxy server MUST reject the request with an error response. The security policy may be violated because a PRMB cannot be viewed, or because a PRMB is not present in the message.

If the request contains encrypted data that the proxy cannot decrypt, the proxy MUST assume that any missing PRMB is within the encrypted data and MUST reject the message with a new error response, 496 (Proxy Undecipherable). The proxy's public key certificate and the Content-Type of the PRMB SHOULD be included in the error response. If more than one PRMB is missing, the proxy MAY use the Content-Type of any missing PRMB in the error response. The proxy's public key certificate SHOULD be set as an "application/pkix-cert" [6] MIME body in the error response.

> If the proxy is able to decrypt all of the encrypted data in the request, but a PRMB is missing, the proxy MUST behave as if a a completely unencrypted request had been sent without the PRMB. If the message is a request, it MAY be rejected with a 403 (Forbidden) response. If it is a response, any existing dialog

MAY be terminated.

The Content-Type that the proxy server needs to view MUST be set in the Warning header with a new warn-code, 380, except when the proxy server needs to view the whole body. If no Warning header specifying Content-Type is set, it indicates that the proxy server needs to view the whole body, not specific content. For example, when the proxy server needs to view a SDP, the following Warning header:

Warning: 380 example.com "Required to View Content-Type 'application/sdp'"

If the proxy server requires to view SDP and several sensitive headers that are hidden in tunneling encryption data as described in Section 23.4.3 of [1], the proxy server MAY respond with the Warning header containing 'message/sip'. Instead of specifying Content-Type, the proxy server MAY respond with no Warning header in order to require to view the whole body including sensitive headers.

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.

In a previous version of this document, 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. When the attached signature just to the whole body is required, this error response MUST contain no Warning header to specify Content-Type that is required signature. When the attached signature to tunneling

SIP message is required as described in Section 23.4 of [1], this error response MAY contain the Warning header with 380 warn-code, specifying 'message/sip', or 'message/sipfrag' Content-Type. The proxy server MAY attach the signature to a "message/sipfrag" [13] 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.

When a UA receives the 495 error response and the confidentiality is needed, the UA MUST recognize the error requiring the signature for the data prior to the encryption.

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 that implement end-to-middle security.

5.1. UAC Behavior

When a UAC sends a SIP request, such as an INVITE request including encrypted message body for end-to-middle confidentiality, it MUST generate S/MIME CMS EnvelopedData, and SHOULD specify the hostname of destined proxy server and Content-ID of the S/MIME CMS EnvelopedData which is to be decrypted by the proxy server in the "Proxy-Inspect-Body" header.

If the UAC decides to share the message body with the UAS and the proxy server that requires the inspection of the message body, the UAC MUST list encrypted CEK with the proxy server's KEK and encrypted CEK with the UAS's KEK at the "recipientInfos" of the CMS EnvelopedData. If the UAC decides to set the message body

separately, the UAC MUST structure a "multipart/mixed" body that contains two CMS EnvelopedData: one encrypted for the UAS and another encrypted for the proxy server. The UAC 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 the UAS. 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].

If the UAC sends a SIP request including encrypted body just for end-to-end, being unaware of the service provided by the proxy server that requires the inspection of the message body, the UAC will get a 496 (Proxy Undecipherable) error response with the public key of the proxy server. The error response MAY contain the Warning header requiring the disclosure of a specific content, or no Warning header.

By obtaining the error response that the Warning header specifies content, the UAC learns that the proxy server requires the disclosure of a specific message body. If the error response contains no Warning header, the UAC learns the proxy server require the disclosure of the whole message body. If the UAC decides to meet the requirement of the proxy server, the UAC MUST generates CMS EnvelopedData and MUST set the "Proxy-Inspect-Body" header as described above. If the UAC decides to share the message body with the UAS and the proxy server, the UAC MUST update the "recipientInfos" of the previous request by adding encrypted CEK with the proxy server's KEK obtained from the error response. If the UAC decides to set the message body separately for the proxy server, the UAC MUST structure a "multipart/mixed" body by adding the CMS EnvelopedData for the proxy server.

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

If the UAC sends a SIP request without the signature, being unaware of the proxy server's service that requires the verification of the message body, the UAC will get a 495 (Signature Required) error response with no Warning header requiring Content-Type.

By obtaining the 495 error response, the UAC learns that an entity in the signaling path, such as the proxy server, requires the signature of the whole message body. If the UAC decides to meet the requirement and has its own public key, the UAC MUST generate the CMS SignedData to attach a signature by computing with its own private

key.

When the UAC 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. The UAC MUST specify the hostname of the proxy server and the Content-ID of the CMS EnvelopedData destined for the proxy server in the "Proxy-Inspect-Body". The UAC also MAY specify the Content-ID of the CMS SignedData following the Content-ID of the CMS EnvelopedData in the header.

For example, if the UAC needs the confidentiality of the SDP, and it knows that the destined proxy server needs to view the both SDPs in a request and the response, the UAC MAY use the CEK reuse mechanism [14][15]. The UAC 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 3.

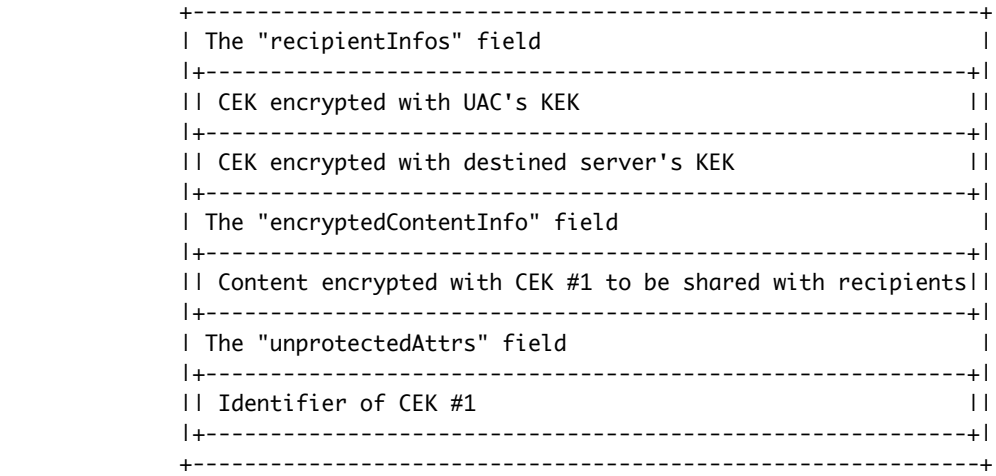


Figure 3: EnvelopedData with CEK reuse in a request

5.2. UAS Behavior

When the UAS receives a request that contains a MIME body, the UAS inspects the MIME body depending on the value of the handling parameter in "Content-Disposition" header. If the MIME body structures S/MIME, the UAS first decrypts and/or validates it as usual. If the decryption and/or the validation is successful, the UAS responds with a 200 OK. If the 200 OK response contains a MIME body, the UAS is RECOMMENDED to construct the same S/MIME structure with the request.

When the CMS EnvelopedData body of the request, destined for the UAS, contains the "unprotectedAttrs" attribute specifying the identifier of the CEK, the UAS MAY learn that the UAC is requesting to reuse the CEK for the disclosure of the message body in the subsequent requests or responses. By checking the "Proxy-Inspect-Body" header in the receiving request, the UAS MAY know the destined proxy server and the Content-Type to be disclosed. If the UAS accepts the disclosure, it MAY keep the CEK with the identifier specified in the "unprotectedAttrs" attribute. If the UAS receives an INVITE message specifying the CEK reuse, the UAS MAY reuse the CEK (CEK #1) to encrypt a new CEK (CEK #2) for encrypting the message body in the response as showed in Figure 4

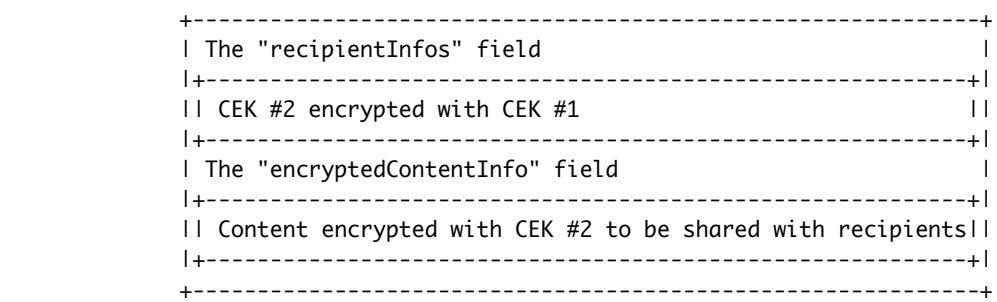


Figure 4: EnvelopedData with CEK reuse in a response

Even when the UAS receives a request that does not use S/MIME, the UAS 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 the UAC operates as described in Section 5.1, while the behavior for discovering the security policies of destined proxy server can not be not supported.

5.3. Proxy Behavior

When the proxy server implementing and activating this mechanism receives a message, it MUST inspect the "Proxy-Inspect-Body" header(s). If the header includes the host name of the proxy server , the proxy server MUST inspect the body indicated by the "cid" parameter. If multiple "cid" parameters exist in the header, the proxy server MUST inspect the bodies in order. Even if the header does not include the hostname of the proxy server, nor the header exists, the proxy server MAY view the message body following its own security policies.

The proxy server MAY activate this mechanism user by user. When the proxy server receives a message from a user for whom the proxy server does not activate this, the proxy server ignores the "Proxy-Inspect-Body" header(s) and does not view the message body.

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

If the proxy server 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 the proxy server requires the disclosure of a specific content, the 496 response MUST include the Warning header, containing "Required to view 'Content-Type'". If the proxy server requires the disclosure of the whole message body, it MUST respond without the Warning header containing Content-Type.

If the proxy server succeeds in this decryption, it MAY inspect the "unprotectedAttrs" field of the CMS EnvelopedData body. If the attribute gives the key's identifier, the proxy server 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, the proxy server MUST validate the digital signature. If the verification fails, the proxy server 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 the proxy server needs validate the data integrity of the content but the indicated body does not contain CMS SignedData body, the proxy server MUST respond with a 495 (Signature Required) response if the message is a request, otherwise any existing dialog MAY be terminated. A 495 response MAY contain no Warning header requiring Content-Type to be attached a signature.

When the proxy server needs to validate the data integrity of the content and view it, but the indicated content is the CMS EnvelopedData, the proxy server does not see if the signature exists inside. First, the proxy server tries to decrypt the CMS EnvelopedData. If the decryption fails, the proxy server 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, the proxy server inspects the content and validate the signature, if it exists. If the signature for the whole body does not exist, the proxy server 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, the proxy server MAY first validate the signature, instead of checking the existence of the signature inside.

When the proxy server forwards the request, it MAY delete the "Proxy-Inspect-Body" header that contains its own hostname.

When a provider operating the proxy server does not allow any information related to its security policies to be revealed to other proxy server, the proxy server MAY deletes the "Proxy-Inspect-Body" header. However, there is no way to conceal the header from the other proxy servers which exist between a UAC and the proxy server.

6. Proxy-Inspect-Body Header

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

```
Proxy-Inspect-Body   = "Proxy-Inspect-Body" HCOLON inspecting-proxy
                       SEMI target-body *(SEMI generic-param)
inspecting-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*( alphanum / "-" / "!" / "%" / "*" /
                      "_" / "+" / "'" / "`" / "~"   )
```

Information about the use of headers in relation to SIP methods and proxy processing is summarized in Table 1.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	REF
Proxy-Inspect-Body	R	dr	o	o	-	o	o	o	o
Proxy-Inspect-Body	100-699	dr	-	o	-	o	o	o	o

Header field	where	proxy	SUB	NOT	PRK	IFO	UPD	MSG	PUB
Proxy-Inspect-Body	R	dr	o	o	o	o	o	o	o
Proxy-Inspect-Body	100-699	dr	o	o	-	o	o	o	o

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:

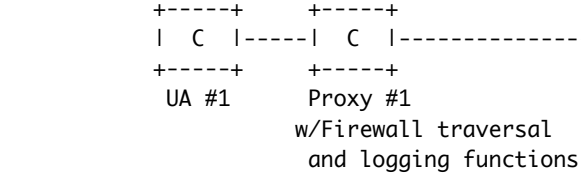
* d: A proxy can delete a header field value.
* r: A proxy must be able to read the header field, so it cannot be encrypted.

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

We describe here the message examples 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 7. The instant messages assumes to use MESSAGE [16] requests.



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

Figure 7: Configuration example

In the message examples, the text with the box of asterisks ("*") is encrypted. Although the Content-Length has no digit, the appropriate length is to be set. The hostname and username for each entity are:

UA #1: alice@atlanta.example.com
UA #2: bob@biloxi.example.com
Proxy #1: ss1.atlanta.example.com
Proxy #2: ss1.biloxi.example.com

7.1. Message Examples of End-to-Middle Confidentiality

In the following example, UA #1 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 --> ss1.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-Inspect-Body: ss1.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 ss1.atlanta.example.com public key            *
* RecipientInfo[1] for Bob's public key                              *
*                                                                    *
* (encryptedContentInfo)                                             *
* Content-Type: application/sdp                                      *
* Content-Length:                                                    *
*                                                                    *
* 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 Proxy #1 and UA #2 successfully views the SDP, UA #2 responds

with a 200 OK. The 200 OK is to be encrypted as follows:

200 OK alice@atlanta.example.com <-- ss1.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=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
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
Content-Length:

*****************************************************************
* (recipientInfos)                                              *
* RecipientInfo[0] for Alice's public key                       *
*                                                               *
* (encryptedContentInfo)                                        *
* Content-Type: application/sdp                                 *
* Content-Length:                                               *
*                                                               *
* 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                                          *
*****************************************************************
```

When keying materials, such as keys used for Secure RTP (SRTP), are included in the SDP [17], 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. In this case, UA #1 sets two SDP separately; one contains the keying materials for UA #2 and another does not for Proxy #1 as follows:

INVITE alice@atlanta.example.com --> ss1.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-Inspect-Body: ss1.atlanta.example.com;
                    cid=1234@atlanta.example.com,cid=3333@atlanta.example.com

Content-Type: multipart/mixed; boundary=boundary1
Content-Transfer-Encoding: binary
Content-ID: 1234@atlanta.example.com
Content-Disposition: attachment;filename=smime.p7m;handling=required
Content-Length:

--boundary1
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
              name=smime.p7m
Content-ID: 3333@atlanta.example.com
Content-Disposition: attachment;filename=smime.p7m;handling=required
*****************************************************************
* (recipientInfos)                                              *
* RecipientInfo[0] for ss1.atlanta.example.com public key       *
*                                                               *
* (encryptedContentInfo)                                        *
* Content-Type: application/sdp                                 *
* Content-Length:                                               *
*                                                               *
* 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/SAVP 0                                      *
*                                                               *
*****************************************************************

--boundary1
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
              name=smime.p7m
Content-ID: 4444@atlanta.example.com
Content-Disposition: attachment;filename=smime.p7m;handling=optional
*****************************************************************
* (recipientInfos)                                              *
* RecipientInfo[0] for Bob's public key                         *
*                                                               *
```

```
* (encryptedContentInfo)                                          *
* Content-Type: application/sdp                                   *
* Content-Length:                                                 *
*                                                                 *
* 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/SAVP 0                                        *
* a=crypto:1 AES_CM_128_HMAC_SHA1_32                              *
*   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32     *
*                                                                 *
*******************************************************************
--boundary1--
```

For firewall traversal service, Proxy #1 does not care about the information only for UA #2. Even if UA #1 sets different port information for UA #2 and Proxy #1 separately on purpose, the firewall traversal service for UA #1 will just fail.

However, if Proxy #1 provides a call admission control using codec information in SDP, Proxy #1 might care about the information for UA #2. Proxy #1 wants 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. As a result, Proxy #1 will require disclosure of all the message body by setting no Warning header requiring Content-Type as follows:

496 Proxy Undecipherable alice@atlanta.example.com <-- ss1.atlanta.example.com

```
SIP/2.0 496 Proxy Undeciperable
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=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Type: application/pkix-cert
Content-Length:

<certificate>
```

In the following example, UA #1 needs message content in a MESSAGE request to be confidential and it allows Proxy #1 to view the message body.

```
MESSAGE alice@atlanta.example.com --> ss1.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-Inspect-Body: ss1.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 ss1.atlanta.example.com public key       *
* RecipientInfo[1] for Bob's public key                         *
*                                                               *
* (encryptedContentInfo)                                        *
* Content-Type: text/plain                                      *
* Content-Length: ...                                           *
*                                                               *
* Hello.                                                        *
* This is confidential.                                         *
*                                                               *
*****************************************************************
```

If Proxy #1 and UA #2 successfully view the message body, UA #1 receives a 200 OK from UA #2 normally. However, if Proxy #1 fails to view the message body, UA #1 receives a 496 (Proxy Undecipherable) error response from Proxy #1, as follows:

```
   496 Proxy Undecipherable alice@atlanta.example.com <--
   ss1.atlanta.example.com


   SIP/2.0 496 Proxy Undeciperable
   Warning: 380 ss1.atlanta.example.com "Required to view 'text/plain'"
   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=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 MESSAGE
   Content-Type: application/pkix-cert
   Content-Length: ...

   <certificate>
```

7.2. Message Examples of End-to-Middle Integrity

In the following example, UA #1 needs the integrity of message content in a MESSAGE request to be validated by Proxy #1 before it views message content.

```
   MESSAGE alice@atlanta.example.com --> ss1.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
   Content-Type: multipart/signed;protocol="application/pkcs7-signature"
            ;micalg=sha1;boundary=boundary1
   Content-Length: ...

   --boundary1
   Content-Type: text/plain
   Content-Length: ...

   Hello.
   This is protected with the signature.
   --boundary1
   Content-Type: application/pkcs7-signature; name=smime.p7s
   Content-Transfer-Encoding: binary
   Content-ID:1234@atlanta.example.com
   Content-Disposition: attachment;
   filename=smime.p7s;handling=required

   [binary data]
   --boundary1--
```

   If Proxy #1 successfully validates the integrity of the message body and UA #2 receive it, UA #1 normally receives a 200 OK from UA #2. if Proxy #1 does not receive a signature for the whole message body, UA #1 receives a 495 (Signature Required) error response from UA #1, as follows:

```
   495 Signature Required alice@atlanta.example.com <--
   ss1.atlanta.example.com


   SIP/2.0 495 Signature Required
   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=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 MESSAGE
   Content-Length: 0
```

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 contains the public key certificate of a proper proxy server, UA MUST see if the common name of the public key certificate is equal to the name of a proper proxy server. UA MUST recognize the name of legitimate proxies by the domain part of the SIP URIs of the UA and recipient, the name of configured outbound proxy and pre-configuration. That is, UA MUST be pre-configured with the domain name of a third party, if there are a third party proxy which offers some service viewing the message body.

Additionally, a UA MUST verify whether the public key certificate is valid and not revoked as specified in [8]. If it is invalid and revoked, the UA MUST NOT retry to send the message with data encrypted by the public key.

If a malicious user sends the public key certificate of the proper proxy server, it would not be a problem as long as the malicious user does not know the corresponding private key, since the malicious user will fail to decrypt the data the user sent.

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 MUST 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 in the middle modified/added/deleted the label, the specified message body will be not inspected by the specified proxy server, and some services requiring its content can not 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-Inspect-Body' Header

This section includes the registration information for the 'Proxy-Inspect-Body' header which is described in Section 6 of this document.

Header Name: Proxy-Inspect-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

Changed from -05.
- Fixed the misuse of "acknowledge", to "learn" in Section 5.1 and 5.2.
- Clarified how to recognize legitimate proxy name in Section 8.1.
- Fixed the problem condition in Section 8.1.

Changed from -04.
- Changed the header name "Proxy-Required-Body" to "Proxy-Inspect-Body".
- Changed the constraint for labeling encrypted data from "SHOULD" to "MUST".
- Changed the constraint for verifying certificate at the recipient from "SHOULD" to "MUST" in Section 8.2.
- Removed the necessity of authentication in Section 8.3.
- Added text for clarification.

Changed from -03.

Added text mentioning CMS Digest-Data.
Added text related the order of sign and encryption in Section 2.3.
Added an example of Warning header in Section 4.1.
Added how to support SIP tunneling sign and encryption in Section 4.1.
Removed examples from the Section 5 "Behavior of UA and Proxy Servers". Then, merged them to Section 7 "Message Examples".
Added message examples with multipart/mixed structure.
Referred to RFC3280 to validate certificated in Section 8.1.
Added the necessity of authentication in Section 8.3

Changes from -02.

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

Changes from -01.

- Changed an author's contact address.

Changes from -00.

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

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Ono, K. and S. Tachimoto, "Requirements for End-to-Middle Security for the Session Initiation Protocol (SIP)", RFC 4189, October 2005.

[3] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling", RFC 3850, July 2004.

[4] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.

[5] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[6] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, May 1999.

[7] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[8] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

12.2. Informative References

[9] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[10] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

[11] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[12] Hilt, V., Camarillo, G., and J. Rosenberg, "Session Initiation Protocol (SIP) Session Policies - Document Format and Session-Independent Delivery Mechanism", draft-ietf-sipping-session-indep-policy-02 (work in progress), February 2005.

[13] Sparks, R., "Internet Media Type message/sipfrag", RFC 3420, November 2002.

[14] Farrell, S. and S. Turner, "Reuse of CMS Content Encryption Keys", RFC 3185, October 2001.

[15] Ono, K. and S. Tachimoto, "Key reuse in S/MIME for SIP", draft-ono-sipping-keyreuse-smime-00 (work in progress), February 2004.

[16] Campbell, Ed., B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[17] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol Security Descriptions for Media Streams", draft-ietf-mmusic-sdescriptions-11 (work in progress), June 2005.

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

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).