SIP
Internet-Draft
Expires: January 16, 2005

C. Jennings
Cisco Systems
J. Peterson
NeuStar, Inc.
July 18, 2004

Certificate Management Service for SIP
draft-jennings-sipping-certs-04

Status of this Memo

Copyright Notice

Abstract

This draft defines a Credential Service that uses a SIP subscribe/ notify mechanism to discover other users' certificates and credentials and be notified about changes to these certificates. Other user agents that want to contact that AOR can retrieve these certificates from the server. The result is that widespread deployment of S/MIME in SIP is possible, because no extra expense or effort is required of the end user.

This work is being discussed on the sipping@ietf.org mailing list.

Table of Contents

1. Introduction

SIP provides a mechanism for end to end encryption and integrity using S/MIME, and several security properties of SIP depend on S/MIME. S/MIME has not been widely implemented or deployed due to the complexity of providing a reasonable key management infrastructure. This document proposes a way to address certificate discovery, retrieval, and management for SIP deployments. It follows the Sacred Framework RFC 3760 [7] for management of the credentials. Combined with the Identity [2] work, this work allows users to have certificates that are not signed by any well known certificate authority while still strongly binding the user's identity to the certificate. This mechanism allows UAs such as IP phones to enroll and get their credentials without any more configuration information than they commonly have today, without any extra effort or key clicks by the end user, and without any extra expense for the end user. This mechanism also lets the UA discover and retrieve the public certificate for any other user and find out about certificate revocations.

The general approach is to provide a new SIP service referred to as a Credential Server that allows UAs to subscribe to some other user's certificate. The certificate is delivered in a SIP NOTIFY to the UA that subscribes. The identity of the certificate can be vouched for using Identity [2] work. The Credential Service can manage public certificates as well as credentials that include the user's private key. The user can install new credentials to the Credential Server using a SIP PUBLISH. The Credential Server authenticates UAs that are changing credentials or requesting private keys using a shared secret that both the UA and the Server know. Typically this will be the same shared secret that is used in Register with the Registrar for the domain.

The mechanism described in this document works for both self signed certificates and certificates signed by a well known certificate

authority; however, it is imagined that most UAs using this would only use self signed certificates and would use an Authentication Service as described in [2] to provide strong identity binding to the certificates.

Previous versions of this draft (00 to 02) used HTTP instead of SIP for communicating with the Credential Server. The key difference with using SIP is that a certificate can be revoked by sending a new NOTIFY; in the HTTP based scheme, the certificates were cached for a predefined period of time, typically one day, so that a revocation could only take effect after the cache expired. The earlier version also did not deal with the SACRED problem and allowed several devices with the same AOR to all have different private keys. This resulted

in very large SIP message and was looking fairly unwieldy; so now, the UAs for one AOR share private keying material and use the SACRED framework to move it between devices.

This basic approach of this work is independent of the details of body modification [13] and identity discussions. However, the choices made there will affect the mechanisms used to implement the approach described here.

2. Conventions

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 [5].

Certificate: An X.509 style certificate containing a public key and a list of identities in the SubjectAltName that are bound to this key. The certificates discussed in this draft are generally self signed and use the mechanisms in the Identity work [2] to vouch for their validity.

Credential: For this document, this means the combination of a certificate and the associated private key.

3. Goals

- Allow negotiation of E2E encrypted sessions.
- Allow end to end encryption and integrity of SIP bodies that may

be delivered in SIP signaling, such as page mode MESSAGEs or NOTIFY bodies in presence.

- o Work for users with multiple UA devices.
- o Provide a certificate revocation mechanism.

4. UA Discovering Certificates

UAs discover certificates by sending a SUBSCRIBE with an event type of pkix-cert to the AOR for which a certificate is desired. This could be a SIP or tel URL. The resulting NOTIFY will contain an application/pkix-cert body which contains the certificates. The UA MUST follow the procedures in Section 11.1 to decide if the received certificate can be used. The UA needs to cache this certificate for future use. The certificate MUST be removed from the cache if it has expired, if it is updated by a subsequent NOTIFY, or if the subscription has been terminated. The NOTIFY containing a certificate must be signed by an Authentication Service as described in Identity. If the identity asserted by the Authentication Service does not match the identity requests, the certificates in the NOTIFY are discarded and MUST NOT be used.

5. UA Discovering and Publishing Credentials

UAs discover credentials by subscribing to their AOR with an event type of credential, which will result in a message containing both an application/pkix-cert body and an application/pkcs8 body that has the associated private key information for the certificate. The UA can change the user's certificate and private key by sending the server a PUBLISH[3] with an event type of credential that contains both an application/pkix-cert and an application/pkcs8 body.

The UA needs to authenticate to the Credential Server for these operations. The UA MUST use TLS to connect to the server. The UA may be configured with a specific name for the Credential Server; otherwise it defaults to the name of the domain in the User's AOR. The TLS connection MUST present a certificate that matches the expected name for the credential server, so that the UA knows it is talking to the correct server. If the certificate presented by the server does not match the expected server, the UA MUST terminate the connection and notify the user. If the UA does not do so, it may end up publishing its private key information to an attacker. The Credential Server will authenticate the UA using the usual SIP Digest

mechanism, so the UA can expect to receive a SIP challenge to the SUBSCRIBE or PUBLISH messages.

The application/pkix-cert body is a DER encoded X.509 certificate [10]. The application/pkcs8 bodies contains a DER encoded PKCS #8 object that contains the private key. The PKCS #8 objects MUST be of type PrivateKeyInfo. The integrity and confidentiality of the PKCS #8 objects is provided by the TLS transport. The transport encoding of all the MIME bodies is binary.

6. Credential Server Behavior

The Credential Server stores credentials for users and can provide the credentials or certificates to other user agents. The credentials are indexed by an URI that corresponds to the AOR of the user. When a UA requests a public certificate with a SUBSCRIBE, the server sends it in a NOTIFY and sends a subsequent NOTIFY any time it changes. When a credential is requested, the Server digest challenges the requesting UA to authenticate it so that the Server can verify that the UA is authorized to receive the requested credentials.

When the Credential Server receives a SUBSCRIBE for a certificate, it first checks to see if it has credentials for the requested URI. If it does not it returns a response indicating the user was not found. Otherwise it sets up a subscription and forms a NOTIFY with the certificate in the body and the From header field value set to the

request URI of the SUBSCRIBE. It MUST send this NOTIFY through an Authentication Service (as described in Identity [2]) or implement an Authentication Service itself. The Server is encouraged to keep the subscriptions active for AORs that are communicating frequently but MAY unsubscribe at any point of time. Any time the credentials for this URI change, the Server MUST send a new NOTIFY to any active subscriptions.

When a Credential Server receives a SUBSCRIBE for a credential, the Server has to authenticate and authorize the UA and validate that adequate transport security is being used. The Server MUST digest challenge the UA to authenticate the UA and then decide if it is authorized to receive the credentials.

Once the UA has authenticated with the Server, the Server can set up a subscription and send a Notify message that MUST contain the credentials. This NOTIFY message is sent thought an Authorization Service in the same way as the certificate subscriptions. If the credential changes, the Server MUST terminate any current subscriptions and force the UA to re-authenticate. This is so that if a secret for retrieving the credentials gets compromised, the rogue UA will not continue to receive credentials after the compromised secret has been changed.

When the Credential Server receives a PUBLISH to update credentials, it MUST authenticate and authorize this the same way it does the subscriptions for credentials. If this succeeds, the Server updates the credential for this URI and processes all the active subscriptions to this URI as described above.

7. Negotiation of Secure Session

SIP uses an offer/answer negotiation mechanism[16] that describes sessions using SDP that may contain keying material, described in [14], for media protocols such as SRTP [15]. This keying material needs to be protected, and SIP does this by encrypting the SDP bodies using S/MIME.

If a UA receives both an unencrypted and an encrypted SDP offer in an multipart/alternative body, it interprets these as it would a normal multipart alternative as defined in RFC 2046 [17], which means it picks the last alternative that it can support. Any bodies that cannot be decrypted are treated as unsupportable. The sending UA should generally put encrypted offers after unencrypted ones, since encrypted ones are preferred. The UA constructs the answer to the offer as it normally would and may include both encrypted and unencrypted versions of the answer using multipart/alternative. The only wrinkle here is that if the UA sent multiple bodies with an

offer, it needs to be able to match the answer (or answeres) to the offer that was chosen.

The UA that made the offer can uniquely identify the various MIME bodies using a MIME Content-ID header. However, the UA sending the answers needs to provide the label of the Content-ID in the response. Solutions were considered that put the Content-ID identifier in a SIP

Header, a MIME header, or an SDP attribute. Since the issue here is fundamentally about providing information that is all at the MIME level about the relation between one set of multipart/alternatives and the other MIME body that is being sent, the best solution seems to involve passing this tag at the MIME level. A new MIME header called "Content-Related-To" updates RFC 2045 with:

```
 rid := "Content-Related-To" ":" msg-id
```

and adds "[rid CRLF]" to the entity-headers.

The identifier supplied in the Content-Related-To header must be a valid Content-ID from a previous MIME message that this body is related to.

The UA looks at the multipart/alternatives and selects the best one it can use. It MUST include a Content-Related-To in the MIME for the answer that copies the tag from the related Content-ID header of the offer body it has chosen to use.

In a typical call from Alice to Bob, Alice would first subscribe to Bob's certificate. If this worked, then Alice would send an Invite to Bob that contained an RTP session in unencrypted SDP and an SRTP session in encrypted SDP. Bob would select the SRTP session and send an answer with encrypted SDP selecting the SRTP session. Both Alice's and Bob's UAs would indicate to the user that a secure call had been negotiated. Alice and Bob could note that the call was secure and adjust their conversation accordingly.

8. Encrypting Bodies of SIP messages

Applications such as presence and 911 location information result in information with significant privacy requirements being sent in SIP. Particular MIME types may define special meanings when both an encrypted and unencrypted body are received, but, unless otherwise specified, the UA SHOULD use the encrypted version if it can decrypt it, and ignore the unencrypted version. There is no requirement for the two versions to have the same information. For example, a page mode message could have an unencrypted version that said "I'm in the Middle East visiting people" while the encrypted version had much

more sensitive information like "I'm over at Osama's house at 21.25'24"N 39.49'24"E". Depending whether the receiving device can decrypt this or not, a different message gets displayed to the receiving user.

9. Signing Bodies of SIP message

In general, signing messages with self-signed certificates is not that useful unless some other means is used to vouch that the certificate has some meaning. If the Authentication Service is used to do this, then the Authentication Service is providing integrity across all the bodies and binding them with an identity. In this case, the additional signature becomes redundant. Because of this, it is recommended that signing bodies SHOULD NOT be used if the certificate is a self signed certificate.

10. Examples

In all these examples, large parts of the message are omitted to highlight what is relevant to this draft. The lines in the examples that are prefixed by $ represent encrypted blocks of data.

10.1 Encrypted Page Mode IM Message

In this example, Alice sends Bob an encrypted page mode instant message. If Alice does not already have Bob's public key from previous communications, she fetches Bob's public key from Bob's credential server:

```
SUBSCRIBE sip:bob@biloxi.example.com SIP/2.0
...
Event: certificate
```

The credential server responds with the certificate in a NOTIFY.

```
NOTIFY alice@atlanta.example.com  SIP/2.0
Subscription-State: active; expires=7200
....
From: <sip:bob@biloxi.example.com>;tag=1234
Identity: "12dsfsdk2389403823cbed"
Identity-Info: sips:billoxi.example.com
....
Event: certificate
Content-Type: application/pkix-cert

< certificate data >
```

Next Alice sends a SIP MESSAGE message to Bob:

```
MESSAGE sip:bob@biloxi.example.com SIP/2.0
...
Content-Type: application/pkcs7-mime

$ Content-Type: text/plain
$
$ < encrypted version of "Hello" >
```

10.2 SRTP Phone Call

In this example, Alice calls Bob and offers both an RTP and an SRTP session. The SDP for the SRTP session contains the SRTP keying material and is encrypted with S/MIME. If Alice does not already have Bob's public key from previous communications, she fetches Bob's public key from Bob's credential server in the same way as shown in the previous example.

Alice sends an INVITE to Bob that offers two alternative SDP bodies, one of which is encrypted and contains the SRTP keying information. The

```
INVITE sip:bob@biloxi.example.com SIP/2.0
...
Content-Type: multipart/alternative;boundary=boundary

--boundary
Content-ID: 123
Content-Type: application/sdp
Content-Disposition: session

< SDP offer for ordinary RTP only >
--boundary
Content-ID: 456
Content-Type: application/pkcs7-mime
Content-Disposition: session

$ Content-Type: application/sdp
```

```
$
$ < encrypted SDP with key for SRTP >
--boundary
```

If Bob's UA does not have Alice's public key, Bob's UA would fetch it

as shown in the previous example. Assuming that Bob's UA supported encryption, it would select the second alternative offer and construct an appropriate answer. The 200 includes the MIME Content-Related-To header that indicates which alternative MIME body was chosen.

```
200 OK
...
Content-ID: 789
Content-Related-To: 456
Content-Type: application/pkcs7-mime
Content-Disposition: session

$ Content-Type: application/sdp
$
$ < encrypted SDP with key for SRTP >
```

10.3 Setting and Retrieving UA Credentials

When Alice's UA wishes to publish Alice's public and private keys to the Credential Server, it sends a PUBLISH message like the one below. This must be sent over a TLS connection in which the other end of the connection presents a certificate that matches the Credential Server for Alice and digest challenges the message to authenticate her.

```
PUBLISH sip:alice@atlanta.example.com SIP/2.0
...
Content-Type: multipart/mixed;boundary=boundary

--boundary
Content-ID: 123
Content-Type: application/pkix-cert
Content-Disposition: session
```

```
< Public certificate for Alice >
--boundary
Content-ID: 456
Content-Type: application/pkcs8
Content-Disposition: session

< Private Key for Alice >
--boundary
```

If one of Alice's UAs subscribes to the credential event, the UA will be digest challenged, and the NOTIFY will include a body similar to

the one in the PUBLISH section above.

11. Security Considerations

This whole scheme is highly dependent on trusting the operators of the Credential Server and trusting that the Credential Server will not be compromised. The security of all the users will be completely compromised if the Credential Server is compromised.

This work requires the TLS session to be used for communications to the Credential Server. Failing to use TLS or selecting a poor cipher suite (such as NULL encryption) will result in credentials being sent unencrypted over the network and will render the whole system useless. Implementation really must use TLS or there is no point in implementing any of this. In addition, the correct checking of chained certificates as specified in TLS [11] is critical for the client to authenticate the server.

If a particular credential needs to be revoked, the new credential is simply published to the Credential Server. Every device keeping this current in its cache will have a subscription to the credential and will rapidly (order of seconds) be notified and replace its cache. Clients that are not subscribed will subscribe and get the new certificate, so they will not end up using the old invalid certificate.

11.1 Trusting the Identity of a Certificate

When a UA wishes to discover the certificate for

sip:alice@example.com, the UA subscribes to the certificate for alice@example.com and receives a certificate in the body of a SIP Notify message. The term original URI is used to describe the original URI that was subscribed to.

If the certificate is signed by a trusted CA, and one of the names in the SubjectAltName matches the original URI, then this certificate MAY be used but only for exactly the Original URI and not for other identities found in the SubjectAltName. Otherwise, there are several steps the UA MUST perform before using this certificate.

- The From header in the NOTIFY message MUST match the original URI.
- The UA MUST check the Identity header as described in the Identity [2] work to validate that bodies have not been tampered with and that an Authentication Service has validated this From header.
- The UA MUST check the validity time of the certificate and stop using the certificate once it is invalid.
- The certificate MAY have several names in the SubjectAltName but the UA MUST only use this certificate when it needs the certificate for the identity in the Original URI. This means that the certificate should only be indexed in the certificate cache by the value of the original URI, not by the value of all the identities found in the SubjectAltName.

These steps result in a chain of bindings that result in a trusted binding between the original URI and a public key. The Original URI is forced to match the From. The Authentication Service validates that this message did come from the identity claimed in the From and that the bodies and From have not been tampered with. The certificate in the body contains the public key for the identity. Only the UA that can authenticate as this user can tamper with this body, so the owner of the identity can provide a false public key but other users cannot. This chain of assertion from original URI, to From, to body, to public key is critical to the security of the mechanism described in this document. If any of the steps above are not followed, this chain of security will be broken and the system will not work.

11.2 Conformity to the SACRED Framework

This work uses the security design outlined in the SACRED Framework [7]. Specifically, it follows the cTLS architecture described in section 4.2.2 of RFC 3760. The client authenticates the server using

the server's TLS certificate. The server authenticates the client using a SIP digest transaction inside the TLS session. The TLS sessions form a strong session key that is used to protect the credentials being exchanged.

Credential Servers SHOULD implement the server name indication extensions in RFC 3546 [8] and they MUST support a TLS profile of TLS_RSA_WITH_AES_128_CBC_SHA as described in RFC 3268 [9] and a profile of TLS_RSA_WITH_3DES_CBC_SHA.

12. IANA

The MIME Content-Related-To header does not require any IANA actions.

12.1 Certificate Event Package

To: ietf-sip-events@iana.org
Subject: Registration of new SIP event package

Package Name: certificate

Is this registration for a Template Package: No

Published Specification(s): draft-jennings-sipping-certs

Person & email address to contact for further information:
Cullen Jennings <fluffy@cisco.com>

12.2 Credential Event Package

To: ietf-sip-events@iana.org
Subject: Registration of new SIP event package

Package Name: credential

Is this registration for a Template Package: No

Published Specification(s): draft-jennings-sipping-certs

Person & email address to contact for further information:
Cullen Jennings <fluffy@cisco.com>

12.3 PKCS #8

To: ietf-types@iana.org
Subject: Registration of MIME media type application/pkcs8

MIME media type name: application

MIME subtype name: pkcs8

Required parameters: None

Optional parameters: None

Encoding considerations: will be binary for 8-bit transports

Security considerations: Carries a cryptographic private key

Interoperability considerations: None

Published specification: draft-jennings-sipping-certs

Applications which use this media type: Any MIME-complaint transport

Additional information:
  Magic number(s): None
  File extension(s): .p8
  Macintosh File Type Code(s): none

Person & email address to contact for further information:
  Cullen Jennings <fluffy@cisco.com>

Intended usage: COMMON

Author/Change controller:
  Cullen Jennings <fluffy@cisco.com>

13. References

13.1 Normative References

[1] RSA Laboratories, "Private-Key Information Syntax Standard, Version 1.2", PKCS 8, November 1993.

[2] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-02 (work in progress), May 2004.

[3] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", draft-ietf-sip-publish-04 (work in progress), May 2004.

[4] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

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

[6] 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.

[7] Gustafson, D., Just, M. and M. Nystrom, "Securely Available Credentials (SACRED) - Credential Server Framework", RFC 3760, April 2004.

[8] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.

[9] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

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

[11] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

13.2 Informational References

[12] Gutmann, P., "Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP", draft-ietf-pkix-certstore-http-07 (work in progress), May 2004.

[13] Mahy, R., "Pros and Cons of allowing SIP Intermediaries to add MIME bodies", draft-mahy-sipping-body-add-00 (work in progress), July 2004.

[14] Andreasen, F., Baugher, M. and D. Wing, "Session Description Protocol Security Descriptions for Media Streams", draft-ietf-mmusic-sdescriptions-06 (work in progress), July 2004.

[15] Baugher, M., McGrew, D., Naslund, M., Carrara, E. and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[16] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[17] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

Authors' Addresses

Cullen Jennings
Cisco Systems
170 West Tasman Drive
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 902-3341
EMail: fluffy@cisco.com

Jon Peterson
NeuStar, Inc.
1800 Sutter St
Suite 570
Concord, CA 94520
US

Phone: +1 925/363-8720
EMail: jon.peterson@neustar.biz
URI: http://www.neustar.biz/

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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