       Certificate Management Service for The Session Initiation Protocol (SIP)
                         draft-ietf-sip-certs-09

Status of this Memo

   This Internet-Draft is submitted to IETF in full conformance with the
   provisions of BCP 78 and BCP 79. This document may contain material
   from IETF Documents or IETF Contributions published or made publicly
   available before November 10, 2008. The person(s) controlling the
   copyright in some of this material may not have granted the IETF
   Trust the right to allow modifications of such material outside the
   IETF Standards Process. Without obtaining an adequate license from
   the person(s) controlling the copyright in such materials, this
   document may not be modified outside the IETF Standards Process, and
   derivative works of it may not be created outside the IETF Standards
   Process, except to format it for publication as an RFC or to
   translate it into languages other than English.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt.

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html.

   This Internet-Draft will expire on March 12, 2010.

Copyright Notice

Abstract

   This draft defines a Credential Service that allows Session
   Initiation Protocol (SIP) User Agents (UAs) to use a SIP event
   package to discover the certificates of other users. This mechanism
   allows user agents that want to contact a given Address-of-Record
   (AOR) to retrieve that AOR's certificate by subscribing to the
   Credential Service, which returns an authenticated response
   containing that certificate. The Credential Service also allows
   users to store and retrieve their own certificates and private keys.

Table of Contents

1. Introduction

[RFC3261], as ammended by [RFC3853], provides a mechanism for end-to-
end encryption and integrity using S/MIME [RFC3851]. Several
security properties of [RFC3261] depend on S/MIME, and yet it has not
been widely deployed. One reason is the complexity of providing a
reasonable certificate distribution infrastructure. This
specification proposes a way to address discovery, retrieval, and
management of certificates for SIP deployments. Combined with the
SIP Identity [RFC4474] specification, this specification 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.

In addition, this specification provides a mechanism that allows SIP
User Agents such as IP phones to enroll and get their credentials
without any more configuration information than they commonly have
today. The end user expends no extra effort.

2. Definitions

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 [RFC2119].

Certificate: A PKIX [RFC5280] 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 SIP Identity
   [RFC4474] specification to vouch for their validity. Certificates
   that are signed by a certificate authority can also be used with
   all the mechanisms in this draft, but it is expected that they are
   used purely as a key carrier and that their validity is not
   checked.
Credential: For this document, credential means the combination of a
   certificate and the associated private key.
Password Phrase: A password used to encrypt a PKCS#8 private key.

3. Overview

The general approach is to provide a new SIP service referred to as a
"credential service" that allows SIP User Agents (UAs) to subscribe
to other users' certificates using a new SIP event package [RFC3265].
The certificate is delivered to the subscribing UA in a corresponding
SIP NOTIFY request. An Authentication Service as described in the
SIP Identity [RFC4474] specification can be used to vouch for the

identity of the sender of the certificate by using the sender's proxy
domain certificate to sign the NOTIFY request. The Authentication
Service is vouching that the sender is allowed to populate the SIP
From header field value. The sender of the message is vouching that
this is an appropriate certificate for the user identified in the SIP
from header field value. The credential service can manage public
certificates as well as the user's private keys. Users can update
their credentials, as stored on the credential service, using a SIP
PUBLISH [RFC3903] request. The UA authenticates to the credential
service using a shared secret when a UA is updating a credential.
Typically the shared secret will be the same one that is used by the
UA to authenticate a REGISTER request with the Registrar for the
domain (usually with SIP Digest Authentication).

The following figure shows Bob publishing his credentials from one of
his User Agents (e.g. his laptop software client), retrieving his
credentials from another of his User Agents (e.g. his mobile phone),
and then Alice retrieving Bob's certificate and sending a message to
Bob. SIP 200-class responses are omitted from the diagram to make the
figure easier to understand.

```
                  example.com domain
                  ------------------
       Alice        Proxy  Auth   Cred                Bob1  Bob2
         |            |      |       | TLS Handshake    |     |
         |   [ Bob generates   ]     |<--------------------->|
         |   [ credentials and ]     | PUBLISH (credential)  |
         |   [ publishes them  ]     |<----------------------|
         |            |      |       | Digest Challenge      |
         |            |      |       |---------------------->|
         |            |      |       | PUBLISH + Digest      |
         |            |      |       |<----------------------|
         |            |      |       |                       |
         |            |      |       | time passes...        |
         |            |      |       |                       |
         |            |      |       | TLS Handshake         |
         |     [ Bob later gets ]    |<----------------->|
         |     [ back his own   ]    | SUBSCRIBE         |
         |     [ credentials    ]    | (credential)      |
         |     [ at another     ]    |<------------------|
         |     [ User Agent     ]    | SUBSCRIBE+Digest  |
         |            |      |        |<------------------|
         |            |      |        | NOTIFY            |
         |            |      |        |------------------>|
         |            |      |        | Bob Decrypts key  |
         |            |      |        |                   |
         |            |      |        |                   |
         |  SUBSCRIBE (certificate) |     Alice fetches  |
         |---------->|----->|----->|      Bob's cert     |
         |            |      |NOTIFY|                     |
         | NOTIFY+Identity  |<-----|                     |
         |<----------+------|       |    Alice uses cert  |
         |            |      |       |    to encrypt       |
         | MESSAGE    |      |       |    message to Bob   |
         |---------->|------+------+----------------->|
```

Bob's UA (Bob2) does a TLS [RFC5246] handshake with the credential
server to authenticate that the UA is connected to the correct
credential server. Then Bob's UA publishes his newly created or
updated credentials. The credential server digest challenges the UA
to authenticate that the UA knows Bob's shared secret. Once the UA
is authenticated, the credential server stores Bob's credentials.

Another of Bob's User Agents (Bob1) wants to fetch its current
credentials. It does a TLS [RFC5246] handshake with the credential
server to authenticate that the UA is connected to the correct
credential server. Then Bob's UA subscribes for the credentials.
The credential server digest challenges the UA to authenticate that
the UA knows Bob's shared secret. Once the UA is authenticated, the

credential server sends a NOTIFY that contains Bob's credentials.
The private key portion of the credential may have been encrypted
with a secret that only Bob's UA (and not the credential server)
knows. In this case, once Bob's UA decrypts the private key it will
be ready to go. Typically Bob's UA would do this when it first
registered on the network.

Some time later Alice decides that she wishes to discover Bob's
certificate so that she can send him an encrypted message or so that
she can verify the signature on a message from Bob. Alice's UA sends
a SUBSCRIBE message to Bob's AOR. The proxy in Bob's domain routes
this to the credential server via an "authentication service" as
defined in [RFC4474]. The credential server returns a NOTIFY that
contains Bob's public certificate in the body. This is routed
through an authentication service that signs that this message really
can validly claim to be from the AOR "sip:bob@example.com". Alice's
UA receives the certificate and can use it to encrypt a message to
Bob.

It is critical to understand that the only way that Alice can trust
that the certificate really is the one for Bob and that the NOTIFY
has not been spoofed is for Alice to check that the Identity
[RFC4474] header field value is correct.

The mechanism described in this document works for both self signed
certificates and certificates signed by well known certificate
authorities. In order to deploy certificates signed by well known
certificate authorities, certificate authorities would have to
support adding SIP URIs to the subjectAltName of the certificates
they generate. This is something which has been rarely implemented
by commercial certificate authorities. However, most UAs would only
use self signed certificates and would use an Authentication Service
as described in [RFC4474] to provide a strong binding of an AOR to
the certificates.

The mechanisms described in this draft allow for three different
styles of deployment:

1. Deployments where the credential server only stores certificates
    and does not store any private key information. If the
    deployment had users with multiple devices, some other scheme
    (perhaps even manual provisioning) would be used to get the right
    private keys onto all the devices that a user uses.
2. Deployments where the credential server stores certificates and
    also stores an encrypted version of the private keys. The
    credential server would not know or need the password phrase for
    decrypting the private key. The credential server would help
    move the private keys between devices but the user would need to

enter a password phrase on each device to allow that device to
decrypt (and encrypt) the private key information.

3. Deployments where the credential server generates and stores the
certificates and private keys. Deployments such as these may not
use password phrases. Consequently, the private keys are not
encrypted inside the PKCS#8 objects. This style of deployment
would often have the credential server, instead of the devices,
create the credentials.

[4](#). UA Behavior with Certificates

When a User Agent wishes to discover some other user's certificate it
subscribes to the "certificate" SIP event package as described in
[Section 6](#) to get the certificate. While the subscription is active,
if the certificate is updated, the Subscriber will receive the
updated certificate in a notification.

The Subscriber needs to decide how long it is willing to trust that
the certificate it receives is still valid. If the certificate is
revoked before it expires, the Notifier will send a notification with
an empty body to indicate that the certificate is no longer valid.
If the certificate is renewed before it expires, the Notifier will
send a notification with a body containing the new certificate. Note
that the Subscriber might not receive the notification if an attacker
blocks this traffic. The amount of time that the Subscriber caches a
certificate SHOULD be configurable. A default of one day is
RECOMMENDED.

Note that the actual duration of the subscription is unrelated to the
caching time or validity time of the corresponding certificate.
Allowing subscriptions to persist after a certificate is no longer
valid ensures that Subscribers receive the replacement certificate in
a timely fashion. The Notifier could return an immediate
notification with the certificate in response to subscribe and then
immediately terminate subscription, setting the reason parameter to
"probation". The Subscriber will have to periodically poll the
Notifier to verify validity of the certificate.

If the UA uses a cached certificate in a request and receives a 437
(Unsupported Certificate) response, it SHOULD remove the certificate
it used from the cache, attempt to fetch the certificate again. If
the certificate is changed, then the UA SHOULD retry the original
request again with the new certificate. This situation usually
indicates that the certificate was recently updated, and that the
Subscriber has not received a corresponding notification. If the
certificate fetched is the same as the one that was previously in the
cache, then the UA SHOULD NOT try the request again. This situation

can happened when the request was retargeted to a different user than
the original request. The 437 response is defined in [RFC4474].

    Note: A UA that has a presence list MAY want to subscribe to the
    certificates of all the presentities in the list when the UA
    subscribes to their presence, so that when the user wishes to
    contact a presentity, the UA will already have the appropriate
    certificate. Future specifications might consider the possibility
    of retrieving the certificates along with the presence documents.

The details of how a UA deals with receiving encrypted messages is
outside the scope of this specification. It is worth noting that if
Charlie's UAS receives a request that is encrypted to Bob, it would
be valid and legal for that UA to send a 302 redirecting the call to
Bob.

5. UA Behavior with Credentials

UAs discover their own credentials by subscribing to their AOR with
an event type of credential as described in Section 7. After a UA
registers, it SHOULD retrieve its credentials by subscribing to them
as described in Section 6.6.

When a UA discovers its credential, the private key information might
be encrypted with a password phrase. The UA SHOULD request that the
user enter the password phrase on the device, and the UA MAY cache
this password phrase for future use.

There are several different cases in which a UA should generate a new
credential:
o If the UA receives a NOTIFY with no body for the credential
   package.
o If the certificate has expired.
o If the certificate's notAfter date is within the next 600 seconds,
   the UA SHOULD attempt to create replacement credentials. The UA
   does this by waiting a random amount of time between 0 and 300
   seconds. If no new credentials have been received in that time,
   the UA creates new credentials to replace the expiring ones and
   sends them in a PUBLISH request (with a SIP-If-Match header field
   set to the current etag). This makes credential collisions both
   unlikely and harmless.
o If the user of the device has indicated via the user interface
   that they wish to revoke the current certificate and issue a new
   one.
Credentials are created by creating a new key pair which will require
appropriate randomness, and then creating a certificate as described
in Section 9.6. The UA MAY encrypt the private key with a password

   phrase supplied by the user. Next, the UA updates the user's
   credential by sending a PUBLISH [RFC3903] request with the
   credentials or just the certificate as described in Section 7.9.

   If a UA wishes to revoke the existing certificate without publishing
   a new one, it MUST send a PUBLISH with an empty body to the
   credential server.

6. Event Package Formal Definition for "certificate"

6.1. Event Package Name

   This document defines a SIP Event Package as defined in [RFC3265].
   The event-package token name for this package is:

           certificate

6.2. Event Package Parameters

   This package defines the "etag" Event header parameter which is valid
   only in NOTIFY requests. It contains a token which represents the
   SIP etag value at the time the notification was sent. Considering
   how infrequently credentials are updated, this hint is very likely to
   be the correct etag to use in the SIP-If-Match header in a SIP
   PUBLISH request to update the current credentials.

             etag-param = "etag" EQUAL token

6.3. SUBSCRIBE Bodies

   This package does not define any SUBSCRIBE bodies.

6.4. Subscription Duration

   Subscriptions to this event package can range from no time to weeks.
   Subscriptions in days are more typical and are RECOMMENDED. The
   default subscription duration for this event package is one day.

   The credential service is encouraged to keep the subscriptions active
   for AORs that are communicating frequently, but the credential
   service MAY terminate the subscription at any point in time.

6.5. NOTIFY Bodies

   The body of a NOTIFY request for this package MUST either be empty or
   contain an application/pkix-cert body (as defined in [RFC2585]) that
   contains the certificate, unless an Accept header field has

negotiated some other type. The Content-Disposition MUST be set to
"signal" as defined in [RFC3204].

A future extension MAY define other NOTIFY bodies. If no "Accept"
header field is present in the SUBSCRIBE, the body type defined in
this document MUST be assumed.

Implementations which generate large notifications are reminded to
follow the message size restrictions for unreliable transports
articulated in Section 18.1.1 of SIP.

6.6. Subscriber Generation of SUBSCRIBE Requests

A UA discovers a certificate by sending a SUBSCRIBE request with an
event type of "certificate" to the AOR for which a certificate is
desired. In general, the UA stays subscribed to the certificate for
as long as it plans to use and cache the certificate, so that the UA
can be notified about changes or revocations to the certificate.

Subscriber User Agents will typically subscribe to certificate
information for a period of hours or days, and automatically attempt
to re-subscribe just before the subscription is completely expired.

When a user de-registers from a device (logoff, power down of a
mobile device, etc.), subscribers SHOULD unsubscribe by sending a
SUBSCRIBE request with an Expires header field of zero.

6.7. Notifier Processing of SUBSCRIBE Requests

When a SIP credential server receives a SUBSCRIBE request with the
certificate event-type, it is not necessary to authenticate the
subscription request. The Notifier MAY limit the duration of the
subscription to an administrator-defined period of time. The
duration of the subscription does not correspond in any way to the
period for which the certificate will be valid.

When the credential server receives a SUBSCRIBE request for a
certificate, it first checks to see if it has credentials for the
requested URI. If it does not have a certificate, it returns a
NOTIFY request with an empty message body.

6.8. Notifier Generation of NOTIFY Requests

Immediately after a subscription is accepted, the Notifier MUST send
a NOTIFY with the current certificate, or an empty body if no
certificate is available for the target user. In either case it
forms a NOTIFY with the From header field value set to the value of
the To header field in the SUBSCRIBE request. This server sending

the NOTIFY needs either to implement an Authentication Service (as
described in SIP Identity [RFC4474]) or else the server needs to be
set up such that the NOTIFY request will be sent through an
Authentication Service. Sending the NOTIFY request through the
Authentication Service requires the SUBSCRIBE request to have been
routed through the Authentication Service, since the NOTIFY is sent
within the dialog formed by the subscription.

6.9. Subscriber Processing of NOTIFY Requests

The resulting NOTIFY will contain an application/pkix-cert body that
contains the requested certificate. The UA MUST follow the
procedures in Section 9.3 to decide if the received certificate can
be used. The UA needs to cache this certificate for future use. The
maximum length of time it should be cached for is discussed in
Section 9.1. The certificate MUST be removed from the cache if the
certificate has been revoked (if a NOTIFY with an empty body is
received), or if it is updated by a subsequent NOTIFY. The UA MUST
check that the NOTIFY is correctly signed by an Authentication
Service as described in [RFC4474]. If the identity asserted by the
Authentication Service does not match the AOR that the UA subscribed
to, the certificate in the NOTIFY is discarded and MUST NOT be used.

6.10. Handling of Forked Requests

This event package does not permit forked requests. At most one
subscription to this event type is permitted per resource.

6.11. Rate of Notifications

Notifiers SHOULD NOT generate NOTIFY requests more frequently than
once per minute.

6.12. State Agents and Lists

The certificate server described in this section which serves
certificates is a state agent and implementations of the certificate
server MUST be implemented as a state agent.

Implementers MUST NOT use the event list extension [RFC4662] with
this event type. It is not possible to make such an approach work,
because the Authentication service would have to simultaneously
assert several different identities.

6.13. Behavior of a Proxy Server

There are no additional requirements on a SIP Proxy, other than to
transparently forward the SUBSCRIBE and NOTIFY requests as required

in SIP. This specification describes the Proxy, Authentication
service, and credential service as three separate services, but it is
certainly possible to build a single SIP network element that
performs all of these services at the same time.

7. Event Package Formal Definition for "credential"

7.1. Event Package Name

This document defines a SIP Event Package as defined in [RFC3265].
The event-package token name for this package is:

    credential

7.2. Event Package Parameters

This package defines the "etag" Event header parameter which is valid
only in NOTIFY requests. It contains a token which represents the
SIP etag value at the time the notification was sent. Considering
how infrequently credentials are updated, this hint is very likely to
be the correct etag to use in the SIP-If-Match header field in a SIP
PUBLISH request to update the current credentials.

    etag-param = "etag" EQUAL token

7.3. SUBSCRIBE Bodies

This package does not define any SUBSCRIBE bodies.

7.4. Subscription Duration

Subscriptions to this event package can range from hours to one week.
Subscriptions in days are more typical and are RECOMMENDED. The
default subscription duration for this event package is one day.

The credential service SHOULD keep subscriptions active for UAs that
are currently registered.

7.5. NOTIFY Bodies

The NOTIFY MUST contain a multipart/mixed (see [RFC2046]) body that
contains both an application/pkix-cert body with the certificate and
an application/pkcs8 body that has the associated private key
information for the certificate. The Content-Disposition MUST be set
to "signal" as defined in [RFC3204].

A future extension MAY define other NOTIFY bodies. If no "Accept"

header field is present in the SUBSCRIBE, the body type defined in
this document MUST be assumed.

The application/pkix-cert body is a DER encoded X.509v3 certificate
[RFC2585]. The application/pkcs8 body contains a DER-encoded
[RFC5208] 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.

7.6. Subscriber Generation of SUBSCRIBE Requests

A Subscriber User Agent will subscribe to its credential information
for a period of hours or days and will automatically attempt to re-
subscribe before the subscription has completely expired.

The Subscriber SHOULD subscribe to its credentials whenever a new
user becomes associated with the device (a new login). The
subscriber SHOULD also renew its subscription immediately after a
reboot, or when the subscriber's network connectivity has just been
re-established.

The UA needs to authenticate with the credential service for these
operations. The UA MUST use TLS to directly connect to the server
acting as the credential service or to a server that is authoritative
for the domain of the credential service. The UA MUST NOT connect
through an intermediate proxy to the credential service. The UA may
be configured with a specific name for the credential service;
otherwise normal SIP routing is used. As described in RFC 3261, the
TLS connection needs to present a certificate that matches the
expected name of the server to which the connection was formed, so
that the UA knows it is talking to the correct server. Failing to do
this may result in the UA publishing its private key information to
an attacker. The credential service 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 requests.

7.7. Notifier Processing of SUBSCRIBE Requests

When a credential service receives a SUBSCRIBE for a credential, the
credential service has to authenticate and authorize the UA and
validate that adequate transport security is being used. Only a UA
that can authenticate as being able to register as the AOR is
authorized to receive the credentials for that AOR. The credential
Service MUST digest challenge the UA to authenticate the UA and then
decide if it is authorized to receive the credentials. If
authentication is successful, the Notifier MAY limit the duration of
the subscription to an administrator-defined period of time. The

duration of the subscription MUST NOT be larger than the length of
time for which the certificate is still valid. The Expires header
field SHOULD be set so that it is not longer than the notAfter date
in the certificate.

7.8. Notifier Generation of NOTIFY Requests

Once the UA has authenticated with the credential service and the
subscription is accepted, the credential service MUST immediately
send a Notify request. The Notifier SHOULD include the current etag
value in the "etag" Event package parameter in the NOTIFY request.
The Authentication Service is applied to this NOTIFY request in the
same way as the certificate subscriptions. If the credential is
revoked, the credential service MUST terminate any current
subscriptions and force the UA to re-authenticate by sending a NOTIFY
with its Subscription-State header field set to "terminated" and a
reason parameter of "deactivated". (This causes a Subscriber to
retry the subscription immediately.) 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.

Any time the credentials for this URI change, the credential service
MUST send a new NOTIFY to any active subscriptions with the new
credentials.

The notification MUST be sent over TLS so that it is integrity
protected and the TLS needs to be directly connected between the UA
and the credential service with no intermediaries.

7.9. Generation of PUBLISH Requests

A user agent SHOULD be configurable to control whether it publishes
the credential for a user or just the user's certificate.

When publishing just a certificate, the body contains an application/
pkix-cert. When publishing a credential, the body contains a
multipart/mixed containing both an application/pkix-cert and an
application/pkcs8 body.

When the UA sends the PUBLISH [RFC3903] request, it needs to do the
following:
o The UA MUST use TLS to directly connect to the server acting as
   the credential service or to a server that is authoritative for
   the domain of the credential service. The UA MUST NOT connect
   through an intermediate proxy to the credential service.

   o The Expires header field value in the PUBLISH request SHOULD be
      set to match the time for which the certificate is valid.
   o If the certificate includes Basic Constraints, it SHOULD set the
      CA flag to false.

7.10. Notifier Processing of PUBLISH Requests

   When the credential service receives a PUBLISH to update credentials,
   it MUST authenticate and authorize this request the same way as for
   subscriptions for credentials. If the authorization succeeds, then
   the credential service MUST perform the following check on the
   certificate:
   o One of the names in the SubjectAltName of the certificate matches
      the authorized user making the request.
   o The notBefore validity time MUST NOT be in the future.
   o The notAfter validity time MUST be in the future.
   o If a CA Basic Constraint is set in the certificate, it is set to
      false.
   If all of these succeed, the credential service updates the
   credential for this URI, processes all the active certificates and
   credential subscriptions to this URI, and generates a NOTIFY request
   with the new credential or certificate.

   If the Subscriber submits a PUBLISH request with no body, this
   revokes the current credentials and causes all subscriptions to the
   credential package to be deactivated as described in the previous
   section. (Note that subscriptions to the certificate package are NOT
   terminated; each subscriber to the certificate package receives a
   notification with an empty body.)

7.11. Subscriber Processing of NOTIFY Requests

   When the UA receives a valid NOTIFY request, it should replace its
   existing credentials with the new received ones. If the UA cannot
   decrypt the PKCS#8 object, it MUST send a 437 (Unsupported
   Certificate) response. Later if the user provides a new password
   phrase for the private key, the UA can subscribe to the credentials
   again and attempt to decrypt with the new password phrase.

7.12. Handling of Forked Requests

   This event package does not permit forked requests.

7.13. Rate of Notifications

   Notifiers SHOULD NOT generate NOTIFY requests more frequently than
   once per minute.

7.14. State Agents and Lists

The credential server described in this section which serves
credentials is a state agent and implementations of the credential
server MUST be implemented as a state agent.

Implementers MUST NOT use the event list extension [RFC4662] with
this event type.

7.15. Behavior of a Proxy Server

The behavior is identical to behavior described for certificate
subscriptions described in Section 6.13.

8. Examples

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

8.1. Encrypted Page Mode IM Message

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

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

The credential service 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: "NJguAbpmYXjnlxFmlOkumMI+MZXjB2iV/NW5xsFQqzD/p4yiovrJBqhd3T
           ZkegnsmoHryzk9gTBH7Gj/erixEFIf82o3Anmb+CIbrgdl03gGaD6ICvkp
           VqoMXZZjdvSpycyHOhh1cmUx3b9Vr3pZuEh+cB01pbMQ8B1ch++iMjw="
Identity-Info: <https://atlanta.example.com/cert>;alg=rsa-sha1
....
Event: certificate
Content-Type: application/pkix-cert
Content-Disposition: signal

< certificate data >
```

Next, Alice sends a SIP MESSAGE message to Bob and can encrypt the
body using Bob's public key as shown below.

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

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

[8.2](). **Setting and Retrieving UA Credentials**

When Alice's UA wishes to publish Alice's public and private keys to
the credential service, it sends a PUBLISH request like the one
below. This must be sent over a TLS connection directly to the
domain of the credential service. The credential service presents a
certificate where the subjectAltName contains an entry that matches
the domain name in the request line of the PUBLISH request and digest
challenges the request to authenticate her.

```
 PUBLISH sips:alice@atlanta.example.com SIP/2.0
 ...
 Event: credential
 Content-Type: multipart/mixed;boundary=boundary
 Content-Disposition: signal

 --boundary
 Content-ID: 123
 Content-Type: application/pkix-cert


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

 < Private Key for Alice >
 --boundary
```
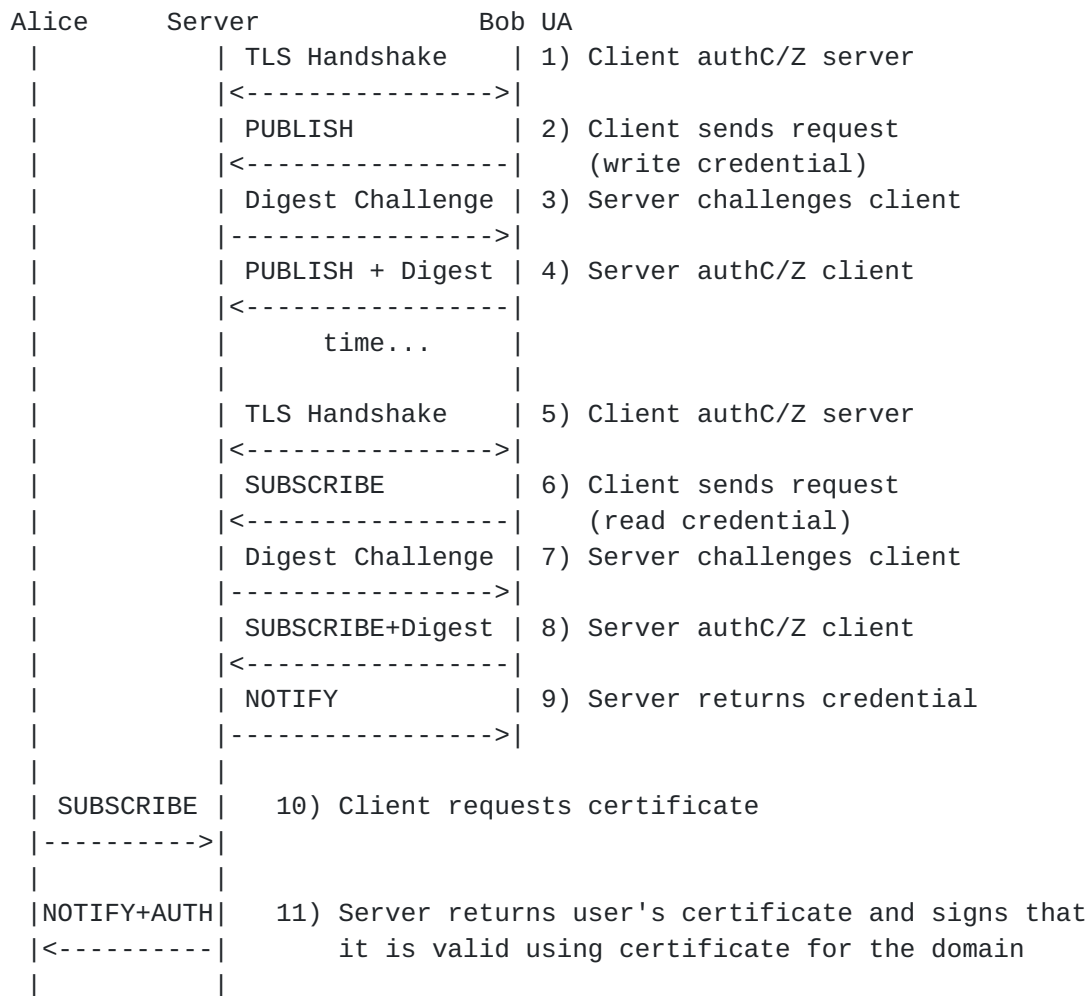
If one of Alice's UAs subscribes to the credential event, the UA will
be digest challenged, and the NOTIFY will include a body similar to
the one in the PUBLISH section above.

9. Security Considerations

The high level message flow from a security point of view is
summarized in the following figure. The 200 responses are removed
from the figure as they do not have much to do with the overall
security.

In this figure, authC refers to authentication and authZ refers to
authorization.

```
Alice        Server              Bob UA
 |              | TLS Handshake    | 1) Client authC/Z server
 |              |<---------------->|
 |              | PUBLISH          | 2) Client sends request
 |              |<----------------|    (write credential)
 |              | Digest Challenge | 3) Server challenges client
 |              |---------------->|
 |              | PUBLISH + Digest | 4) Server authC/Z client
 |              |<----------------|
 |              |      time...     |
 |              |                  |
 |              | TLS Handshake    | 5) Client authC/Z server
 |              |<--------------->|
 |              | SUBSCRIBE        | 6) Client sends request
 |              |<----------------|    (read credential)
 |              | Digest Challenge | 7) Server challenges client
 |              |---------------->|
 |              | SUBSCRIBE+Digest | 8) Server authC/Z client
 |              |<----------------|
 |              | NOTIFY           | 9) Server returns credential
 |              |---------------->|
 |           |
 | SUBSCRIBE |    10) Client requests certificate
 |---------->|
 |           |
 |NOTIFY+AUTH|    11) Server returns user's certificate and signs that
 |<----------|        it is valid using certificate for the domain
 |           |
```

When the UA, labeled Bob, first created a credential for Bob, it
would store this on the credential server. The UA authenticated the
Server using the certificates from the TLS handshake. The Server
authenticated the UA using a digest style challenge with a shared
secret.

The UA, labeled Bob, wishes to request its credentials from the
server. First it forms a TLS connection to the Server, which
provides integrity and privacy protection and also authenticates the

server to Bob's UA. Next the UA requests its credentials using a
SUBSCRIBE request. The Server digest challenges this to authenticate
Bob's UA. The server and Bob's UA have a shared secret that is used
for this. If the authentication is successful, the server sends the
credentials to Bob's UA. The private key in the credentials may have
been encrypted using a shared secret that the server does not know.

A similar process would be used for Bob's UA to publish new
credentials to the server. Bob's UA would send a PUBLISH request
containing the new credentials. When this happened, all the other
UAs that were subscribed to Bob's credentials would receive a NOTIFY
with the new credentials.

Alice wishes to find Bob's certificate and sends a SUBSCRIBE to the
server. The server sends the response in a NOTIFY. This does not
need to be sent over a privacy or integrity protected channel, as the
Authentication service described in [RFC4474] provides integrity
protection of this information and signs it with the certificate for
the domain.

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

   Note: There has been significant discussion of the topic of
   avoiding deployments in which the credential servers store the
   private keys, even in some encrypted form that the credential
   server does not know how to decrypt. Various schemes were
   considered to avoid this but they all result in either moving the
   problem to some other server, which does not seem to make the
   problem any better, or having a different credential for each
   device. For some deployments where each user has only one device
   this is fine but for deployments with multiple devices, it would
   require that when Alice went to contact Bob, Alice would have to
   provide messages encrypted for all of Bob's devices. The sipping
   working group did consider this architecture and decided it was
   not appropriate due both to the information it revealed about the
   devices and users and the amount of signaling required to make it
   work.

This specification requires that TLS be used for the SIP
communications to place and retrieve a UA's private key. This
provides security in two ways:
1. Confidentiality is provided for the digest authentication
    exchange, thus protecting it from dictionary attacks.

2. Confidentiality is provided for the private key, thus protecting
    it from being exposed to passive attackers.
In order to prevent man-in-the-middle attacks, TLS clients MUST check
that the SubjectAltName of the certificate for the server they
connected to exactly matches the server they were trying to connect
to. The connection must be directly connected to the correct server
or any intermediaries in the TLS path can compromise the certificate
and instead provide a certificate for which the attacker knows the
private key. This may lead the UA that relies on this compromised
certificate to lose confidential information. Failing to use TLS or
selecting a poor cipher suite (such as NULL encryption) may result in
credentials, including private keys, being sent unencrypted over the
network and will render the whole system useless.

The correct checking of chained certificates as specified in TLS
[RFC5246] is critical for the client to authenticate the server. If
the client does not authenticate that it is talking to the correct
credential service, a man in the middle attack is possible.

9.1. Certificate Revocation

If a particular credential needs to be revoked, the new credential is
simply published to the credential service. Every device with a copy
of the old credential or certificate in its cache will have a
subscription and will rapidly (order of seconds) be notified and
replace its cache. Clients that are not subscribed will subscribe
when they next need to use the certificate and will get the new
certificate.

It is possible that an attacker could mount a DOS attack such that
the UA that had cached a certificate did not receive the NOTIFY with
its revocation. To protect against this attack, the UA needs to
limit how long it caches certificates. After this time, the UA would
invalidate the cached information even though no NOTIFY had ever been
received due to the attacker blocking it.

The duration of this cached information is in some ways similar to a
device deciding how often to check a CRL list. For many
applications, a default time of 1 day is suggested, but for some
applications it may be desirable to set the time to zero so that no
certificates are cached at all and the credential is checked for
validity every time the certificate is used.

The UA MUST NOT cache the certificates for a period longer than that
of the subscription duration. This is to avoid the UA using invalid
cached credentials when the notifier of the new credentials has been
prevented from updating the UA.

9.2. Certificate Replacement

   The UAs in the system replace the certificates close to the time that
   the certificates would expire. If a UA has used the same key pair to
   encrypt a very large volume of traffic, the UA MAY choose to replace
   the credential with a new one before the normal expiration.

9.3. 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 request. The term original URI is used to describe the URI
   that was in the To header field value of the SUBSCRIBE request. So
   in this case the original URI would be sip:alice@example.com.

   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.
   o The From header field in the NOTIFY request MUST match the
      original URI that was subscribed to.
   o The UA MUST check the Identity header field as described in the
      Identity [RFC4474] specification to validate that bodies have not
      been tampered with and that an Authentication Service has
      validated this From header field.
   o The UA MUST check the validity time of the certificate and stop
      using the certificate if it is invalid. (Implementations are
      reminded to verify both the notBefore and notAfter validity
      times.)
   o 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 asserted by the Authentication
      Service in the NOTIFY. This means that the certificate should
      only be indexed in the certificate cache by the AOR that the
      Authentication Service asserted and not by the value of all the
      identities found in the SubjectAltName list.
   These steps result in a chain of bindings that result in a trusted
   binding between the original AOR that was subscribed to and a public
   key. The original AOR is forced to match the From. The
   Authentication Service validates that this request did come from the
   identity claimed in the From header field value and that the bodies
   in the request that carry the certificate 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 AOR, or devices
   with access to the private key of the domain, can tamper with this
   body. This stops other users from being able to provide a false

public key. 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 specification. If any of the steps above are not
followed, this chain of security will be broken and the system will
not work.

9.4. SACRED Framework

This specification includes a mechanism that allows end users to
share the same credentials across different end-user devices. This
mechanism is based on the one presented in the SACRED Framework
[RFC3760]. While this mechanism is fully described in this document,
the requirements and background are more thoroughly discussed in
[RFC3760].

Specifically, Section 7.6, Section 7.7 and Section 7.10 follow the
cTLS architecture described in section 4.2.2 of [RFC3760]. 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.

9.5. Crypto Profiles

Credential services SHOULD implement the server name indication
extensions in [RFC5246] and they MUST support a TLS profile of
TLS_RSA_WITH_AES_128_CBC_SHA as described in [RFC5246] as a profile
of TLS_RSA_WITH_3DES_EDE_CBC_SHA.

The PKCS#8 in the clients MUST implement PBES2 with a key derivation
algorithm of PBKDF2 using HMAC with SHA1 and an encryption algorithm
of DES-EDE2-CBC-Pad as defined in [RFC2898]. It is RECOMMENDED that
this profile be used when using PKCS#8. A different passphrase
SHOULD be used for the PKCS#8 encryption than is used for server
authentication.

9.6. User Certificate Generation

The certificates should be consistent with [RFC5280]. A
signatureAlgorithm of sha1WithRSAEncryption MUST be implemented. The
Issuers SHOULD be the same as the subject. Given the ease of issuing
new certificates with this system, the Validity can be relatively
short. A Validity of one year or less is RECOMMENDED. The
subjectAltName must have a URI type that is set to the SIP URL
corresponding to the user AOR. It MAY be desirable to put some
randomness into the length of time for which the certificates are
valid so that it does not become necessary to renew all the
certificates in the system at the same time.

It is worth noting that a UA can discover the current time by looking
at the Date header field value in the 200 response to a REGISTER
request.

9.7. Compromised Authentication Service

One of this worst attacks against this system would be if the
Authentication Service were compromised. This attack is somewhat
analogous to a CA being compromised in traditional PKI systems. The
attacker could make a fake certificate for which it knows the private
key, use it to receive any traffic for a given use, and then re-
encrypt that traffic with the correct key and forward the
communication to the intended receiver. The attacker would thus
become a man in the middle in the communications.

There is not too much that can be done to protect against this. A UA
MAY subscribe to its own certificate under some other identity to try
to detect whether the credential server is handing out the correct
certificates. It will be difficult to do this in a way that does not
allow the credential server to recognize the user's UA.

The UA MAY also save the fingerprints of the cached certificates and
warn users when the certificates change significantly before their
expiry date.

The UA MAY also allow the user to see the fingerprints for the cached
certificates so that they can be verified by some other out of band
means.

10. IANA Considerations

This specification defines two new event packages that IANA is
requested to add the registry at:
      http://www.iana.org/assignments/sip-events
It also defines a new mime type that IANA is requested to add to the
registry at:
      http://www.iana.org/assignments/media-types/application

10.1. 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): This document

New Event header parameters: This package defines no
                             new parameters

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

10.2. 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): This document

New Event header parameters: "etag"

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

10.3. 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: binary

   Security considerations: Carries a cryptographic private key

   Interoperability considerations:
        The PKCS#8 object inside this MIME type MUST be DER-encoded

   Published specification:
        RSA Laboratories, "Private-Key Information Syntax Standard,
        Version 1.2", PKCS 8, November 1993.

   Applications which use this media type: Any MIME-compliant 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:
     the IESG

11. Acknowledgments

   Many thanks to Eric Rescorla, Jim Schaad, Rohan Mahy for significant
   help and discussion. Many others provided useful comments, including
   Kumiko Ono, Peter Gutmann, Russ Housley, Yaron Pdut, Aki Niemi,
   Magnus Nystrom, Paul Hoffman, Adina Simu, Dan Wing, Mike Hammer and
   Lyndsay Campbell. Rohan Mahy, John Elwell, and Jonathan Rosenberg
   provided detailed review and text.

12. References

12.1. Normative References

[RFC2046]   Freed, N. and N. Borenstein, "Multipurpose Internet Mail
            Extensions (MIME) Part Two: Media Types", RFC 2046,
            November 1996.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2585]   Housley, R. and P. Hoffman, "Internet X.509 Public Key
            Infrastructure Operational Protocols: FTP and HTTP",
            RFC 2585, May 1999.

[RFC2898]   Kaliski, B., "PKCS #5: Password-Based Cryptography
            Specification Version 2.0", RFC 2898, September 2000.

[RFC3204]   Zimmerer, E., Peterson, J., Vemuri, A., Ong, L., Audet,
            F., Watson, M., and M. Zonoun, "MIME media types for ISUP
            and QSIG Objects", RFC 3204, December 2001.

[RFC3261]   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.

[RFC3265]   Roach, A., "Session Initiation Protocol (SIP)-Specific
            Event Notification", RFC 3265, June 2002.

[RFC3903]   Niemi, A., "Session Initiation Protocol (SIP) Extension
            for Event State Publication", RFC 3903, October 2004.

[RFC4474]   Peterson, J. and C. Jennings, "Enhancements for
            Authenticated Identity Management in the Session
            Initiation Protocol (SIP)", RFC 4474, August 2006.

[RFC5208]   Kaliski, B., "Public-Key Cryptography Standards (PKCS) #8:
            Private-Key Information Syntax Specification Version 1.2",
            RFC 5208, May 2008.

[RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
            (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280]   Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
            Housley, R., and W. Polk, "Internet X.509 Public Key
            Infrastructure Certificate and Certificate Revocation List
            (CRL) Profile", RFC 5280, May 2008.

12.2. Informational References

[RFC3760]   Gustafson, D., Just, M., and M. Nystrom, "Securely
            Available Credentials (SACRED) - Credential Server
            Framework", RFC 3760, April 2004.

[RFC3851]   Ramsdell, B., "Secure/Multipurpose Internet Mail
            Extensions (S/MIME) Version 3.1 Message Specification",
            RFC 3851, July 2004.

[RFC3853]   Peterson, J., "S/MIME Advanced Encryption Standard (AES)
            Requirement for the Session Initiation Protocol (SIP)",
            RFC 3853, July 2004.

[RFC4662]   Roach, A., Campbell, B., and J. Rosenberg, "A Session
            Initiation Protocol (SIP) Event Notification Extension for
            Resource Lists", RFC 4662, August 2006.

Authors' Addresses

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

   Phone: +1 408 421-9990
   Email: fluffy@cisco.com

   Jason Fischl (editor)
   Skype
   2145 Hamilton Ave.
   San Jose, CA 95125
   USA

   Phone: +1-415-202-5192
   Email: jason.fischl@skypelabs.com