

PKIX  
Internet-Draft  
Intended status: Standards Track  
Expires: August 28, 2013

M. Pritikin, Ed.  
Cisco Systems, Inc.  
P. Yee, Ed.  
AKAYLA, Inc.  
D. Harkins, Ed.  
Aruba Networks  
February 24, 2013

Enrollment over Secure Transport  
draft-ietf-pkix-est-05

Abstract

This document profiles certificate enrollment for clients using Certificate Management over CMS (CMC) messages over a secure transport. This profile, called Enrollment over Secure Transport (EST), describes a simple yet functional certificate management protocol targeting Public Key Infrastructure (PKI) clients that need to acquire client certificates and associated Certification Authority (CA) certificate(s). It also supports client-generated public/private key pairs as well as key pairs generated by the CA.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on August 28, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|   |    |
|---|----|
| 1. Introduction . . . . .   | 4  |
| 1.1. Terminology . . . . .  | 5  |
| 2. Operational Scenario Overviews . . . . .                         | 6  |
| 2.1. Obtaining CA Certificates . . . . .                            | 7  |
| 2.2. Initial Enrollment . . . . .                                   | 8  |
| 2.2.1. Certificate TLS authentication . . . . .                     | 8  |
| 2.2.2. Certificate-less TLS authentication . . . . .                | 8  |
| 2.2.3. HTTP-based client authentication . . . . .                   | 9  |
| 2.3. Client Certificate Re-issuance . . . . .                       | 9  |
| 2.4. Server Key Generation . . . . .                                | 9  |
| 2.5. Full PKI Request messages . . . . .                            | 9  |
| 2.6. Certificate Signing Request (CSR) Attributes Request . . . . . | 9  |
| 3. Protocol Design and Layering . . . . .                           | 10 |
| 3.1. Application Layer . . . . .                                    | 14 |
| 3.2. HTTP Layer . . . . .   | 15 |
| 3.2.1. HTTP headers for control . . . . .                           | 15 |
| 3.2.2. HTTP URIs for control . . . . .                              | 16 |
| 3.2.3. HTTP-Based Client Authentication . . . . .                   | 17 |
| 3.2.4. Message types . . . . .                                      | 18 |
| 3.3. TLS Layer . . . . .  | 19 |
| 3.3.1. TLS-Based Server Authentication . . . . .                    | 20 |
| 3.3.2. TLS-Based Client Authentication . . . . .                    | 20 |
| 3.3.3. Certificate-less TLS Mutual Authentication . . . . .         | 21 |
| 3.4. Proof-of-Possession . . . . .                                  | 21 |
| 3.5. Linking Identity and PoP information . . . . .                 | 22 |
| 3.6. Server Authorization . . . . .                                 | 23 |
| 3.6.1. Client use of Explicit TA Database . . . . .                 | 23 |
| 3.6.2. Client use of Implicit TA Database . . . . .                 | 23 |
| 3.7. Client Authorization . . . . .                                 | 23 |
| 4. Protocol Exchange Details . . . . .                              | 24 |
| 4.1. Distribution of CA certificates . . . . .                      | 24 |
| 4.1.1. Bootstrap Distribution of CA certificates . . . . .          | 24 |
| 4.1.2. Distribution of CA certificates request . . . . .            | 25 |
| 4.1.3. Distribution of CA certificates response . . . . .           | 25 |
| 4.2. Client Certificate Request Functions . . . . .                 | 26 |
| 4.2.1. Simple Enrollment of Clients . . . . .                       | 27 |
| 4.2.2. Simple Re-Enrollment of Clients . . . . .                    | 28 |
| 4.2.3. Simple Enroll and Re-Enroll Response . . . . .               | 28 |

|             |   |    |
|-------------|---|----|
| 4.3.        | Full CMC . . . . .                                  | 29 |
| 4.3.1.      | Full CMC Request . . . . .                          | 29 |
| 4.3.2.      | Full CMC Response . . . . .                         | 29 |
| 4.4.        | Server-side Key Generation . . . . .                | 30 |
| 4.4.1.      | Server-side Key Generation Request . . . . .        | 30 |
| 4.4.2.      | Server-side Key Generation Response . . . . .       | 31 |
| 4.5.        | CSR Attributes . . . . .                            | 32 |
| 4.5.1.      | CSR Attributes Request . . . . .                    | 32 |
| 4.5.2.      | CSR Attributes Response . . . . .                   | 32 |
| 5.          | Contributors/Acknowledgements . . . . .             | 34 |
| 6.          | IANA Considerations . . . . .                       | 34 |
| 7.          | Security Considerations . . . . .                   | 35 |
| 8.          | References . . . . .                                | 37 |
| 8.1.        | Normative References . . . . .                      | 37 |
| 8.2.        | Informative References . . . . .                    | 40 |
| Appendix A. | Operational Scenario Example Messages . . . . .     | 41 |
| A.1.        | Obtaining CA Certificates . . . . .                 | 41 |
| A.2.        | Certificate TLS authentication . . . . .            | 42 |
| A.3.        | Username/Password Distributed Out-of-Band . . . . . | 44 |
| A.4.        | Re-Enrollment . . . . .                             | 47 |
| A.5.        | Server Key Generation . . . . .                     | 48 |
| A.6.        | CSR Attributes . . . . .                            | 52 |
| Authors'    | Addresses . . . . .                                 | 52 |

## 1. Introduction

This document profiles certificate enrollment for clients using Certificate Management over CMS (CMC) [RFC5272] messages over a secure transport. Enrollment over Secure Transport (EST) describes the use of Transport Layer Security (TLS) 1.1 [RFC4346] (or a later version) and Hypertext Transfer Protocol (HTTP) 1.1 [RFC2616] (or a later version) to provide an authenticated and authorized channel for Simple PKI (Public Key Infrastructure) Requests and Responses [RFC5272].

Architecturally, the EST service is located between a CA and a client device. It performs several functions traditionally allocated to the RA (Registration Authority) role in a PKI. The nature of communication between an EST server and a CA is not described in this document.

EST adopts the Certificate Management Protocol (CMP) [RFC4210] model for CA certificate rollover, but does not use the CMP message syntax or protocol. EST servers are extensible in that new functions may be defined to provide additional capabilities not specified in CMC [RFC5272], and this document defines two such extensions, one for requesting Certificate Signing Request attributes, and another for requesting server-generated keys.

EST specifies how to transfer messages securely via HTTP over TLS (HTTPS) [RFC2818], where the HTTP headers and content types are used in conjunction with TLS. HTTPS operates over TCP; this document does not specify EST over Datagram Transport Layer Security/User Datagram Protocol (DTLS/UDP). Figure 1 shows how the layers build upon each other.

EST Layering:

Protocols:

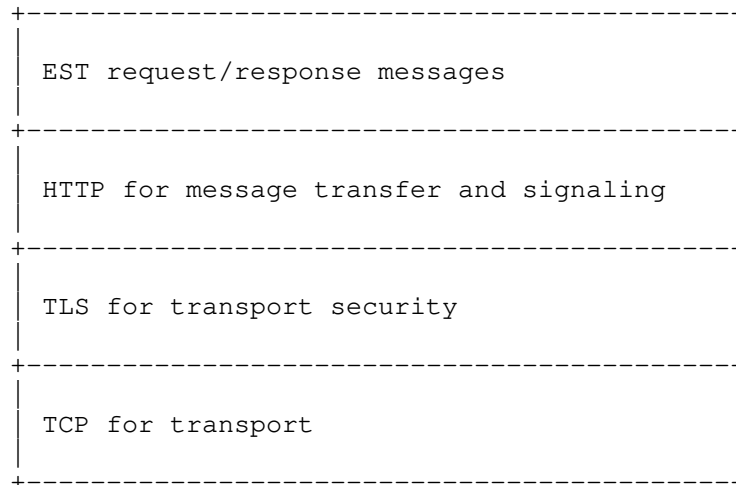


Figure 1

[[EDNOTE: Comments such as this one, included within double brackets and initiated with an 'EDNOTE', are for editorial use and shall be removed as the document is polished.]]

### 1.1. Terminology

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].

It is assumed that the reader is familiar with the terms and concepts described in Public Key Cryptography Standard (PKCS) #10 [RFC2314], HTTPS [RFC2818], CMP [RFC4210], CMC [RFC5272][RFC5273][RFC5274], and TLS [RFC4346].

In addition to the terms defined in the terminology section of CMC [RFC5272] the following terms are defined for clarity:

EST CA: For certificate issuing services, the EST CA is reached through the EST Server; the CA could be logically "behind" the EST Server or embedded within it.

**Third-Party Trust Anchor (TA):** Any Trust Anchor that is not authoritative for the PKI hierarchy the EST server is providing services for.

**Explicit Trust Anchor:** Any Trust Anchor that is explicitly configured on the client for use during EST TLS authentication. For example a TA that is manually configured on the EST client or bootstrapped as described in Section 4.1.1. (See more details in Section 3.6 and Section 7).

**Implicit Trust Anchor:** Any third-party Trust Anchor that is available on the client for use during TLS authentication but is not specifically indicated for use during EST TLS authentication. For example TAs commonly used by web browsers to authenticate web servers. The authorization model for these TAs is different from the authorization model for Explicit Trust Anchors. (See more details in Section 3.6.1, Section 3.6.2 and Section 7).

**Certificate-less TLS:** Use of a TLS cipher suite in which neither the client nor server use a certificate to authenticate. The credential used for authentication is a word, phrase, code or key that is shared between the client and server. The credential must be uniquely shared between the client and server in order to provide authentication of an individual client.

## 2. Operational Scenario Overviews

This section provides an informative overview of the operational scenarios to better introduce the reader to the protocol discussion. This section does not include RFC 2119 key words.

Both the EST clients and server are configured with information that provides the basis for bidirectional authentication and for authorization. The specific initialization data depends on the methods available in the client device and server, but can include shared secrets, network service names and locations (e.g., a Uniform Resource Identifier (URI) [RFC3986]), trust anchor information (e.g., a CA certificate or a hash of a TA's certificate), and enrollment keys and certificates. Depending on an enterprise's acquisition and network management practices, some initialization may be performed by the vendor prior to delivery of client hardware and software. In that case, the client device vendor may provide data, such as trust anchors, to the enterprise via a secure procedure. The distribution of this initial information is out of scope.

Distribution of trust anchors and other certificates can be effected via the EST server. However, nothing can be inferred about the

authenticity of this data until an out-of-band mechanism is used to verify them.

Sections 2.1-2.3 very closely mirror the text of the Scenarios Appendix of [RFC6403] with such modifications as are appropriate for this profile. Sections 2.1-2.6, below, enumerate the set of EST functions (see Figure 5) and provide an informative overview of EST's capabilities.

The general client/server interaction proceeds as follows: The client device initiates a TLS-secured HTTP session with an EST server. A specific EST service is requested based on a portion of the URI used for the session. The client device and server authenticate each other. The client verifies that the server is authorized to serve this client. The server verifies that the client is authorized to make use of this server and the request that the client has made. The server acts upon the client request.

## 2.1. Obtaining CA Certificates

The EST client can request a copy of the current EST CA certificates from the EST server. The EST client is assumed to perform this operation before performing other operations.

Throughout this document we assume the EST CA has a certificate that is used by the client to verify signed objects issued by the CA, e.g., certificates and certificate revocation lists (CRLs), and that a separate end-entity (EE) certificate is used when EST protocol communication requires additional encryption. This operation is used to obtain the EST CA certificate(s).

The EST client authenticates and verifies the authorization scope of the EST server when requesting the current CA certificate(s). As detailed in Section 3.3.1 and Section 3.3.3, available options include:

- o Verifying the EST server's HTTPS URI against the EST server's certificate using Implicit TAs (similar to a common HTTPS exchange). This allows the EST server and client to leverage existing TAs that might be known to the EST client.
- o The client can leverage a previously distributed trust anchor specific to the EST server. This allows the EST client to use an existing, potentially older, CA certificate to request a current CA certificate.
- o For bootstrapping, the EST client can rely upon manual authentication performed by the end user as detailed in

#### Section 4.1.1.

Client authentication is not required for this exchange, so it is trivially supported by the EST server.

### 2.2. Initial Enrollment

After authenticating an EST server and verifying that it is authorized to provide services to the client, an EST client can acquire a certificate for itself by submitting an enrollment request to that server.

The EST server authenticates and authorizes the EST client as specified in Section 3.3.2, Section 3.3.3 and Section 3.7. The methods described in the normative text that are discussed in this overview include:

- o TLS with a previously issued client certificate (e.g., an existing certificate issued by the EST CA);
- o TLS with a previously installed certificate (e.g., manufacturer installed certificate or a certificate issued by some other party);
- o Certificate-less TLS (e.g., with a shared credential distributed out-of-band);
- o HTTP-based with a username/password distributed out-of-band.

#### 2.2.1. Certificate TLS authentication

If the EST client has a previously installed certificate issued by a third party CA, this certificate can be used to authenticate the client's request for a certificate from the EST server (if that CA is recognized by the EST server). An EST client responds to the EST server's TLS certificate request message with the existing certificate already held by the client. The EST server will verify the client's existing certificate and authorize the client's request as described in Section 3.3.2.

#### 2.2.2. Certificate-less TLS authentication

The EST client and EST server can be mutually authenticated using a certificate-less TLS cipher suite (see Section 3.3.3).

### 2.2.3. HTTP-based client authentication

The EST server can optionally also request that the EST client submit a username/password using the HTTP Basic or Digest Authentication methods (see Section 3.2.3). This approach is desirable if the EST client cannot be authenticated during the TLS handshake (see Section 3.3.2) or the EST server policy requires additional authentication information. See Section 3.2.3.

### 2.3. Client Certificate Re-issuance

An EST client can renew/rekey its existing client certificate by submitting a re-enrollment request to an EST server.

When the current EST client certificate can be used for TLS client authentication (Section 3.3.2) the client presents this certificate to the EST server for client authentication. When the to be re-issued EST client certificate cannot be used for TLS client authentication, any of the authentication methods used for initial enrollment can be used.

For example if the client has an alternative certificate issued by the EST CA that can be used for TLS client authentication, then it can be used.

The certificate request message includes the same Subject and SubjectAltName as the current certificate. Name changes are requested as specified in Section 4.2.2.

### 2.4. Server Key Generation

The EST client can request a server-generated certificate and key pair (see Section 4.4).

### 2.5. Full PKI Request messages

Full PKI Request [RFC5272] messages can be transported via EST using the Full CMC Request function. This affords access to functions not provided by the Simple Enrollment functions. Full PKI Request messages are defined in Sections 3.2 and 4.2 of [RFC5272]. See Section 4.3 for a discussion of how EST provides a transport for these messages.

### 2.6. Certificate Signing Request (CSR) Attributes Request

Prior to sending an enrollment request to an EST server, an EST client can query the EST server for a set of additional attribute(s) that the client is requested to use in a subsequent enrollment

request.

These attributes can provide additional descriptive information that the EST server cannot access itself, such as the MAC address of an interface. Alternatively, these attributes can indicate the kind of enrollment request, such as a specific elliptic curve or a specific hash function that the client is expected to use when generating the CSR.

### 3. Protocol Design and Layering

Figure 2 provides an expansion of Figure 1 describing how the layers are used. Each aspect is described in more detail in the sections that follow.

EST Layering:

Protocols and uses:

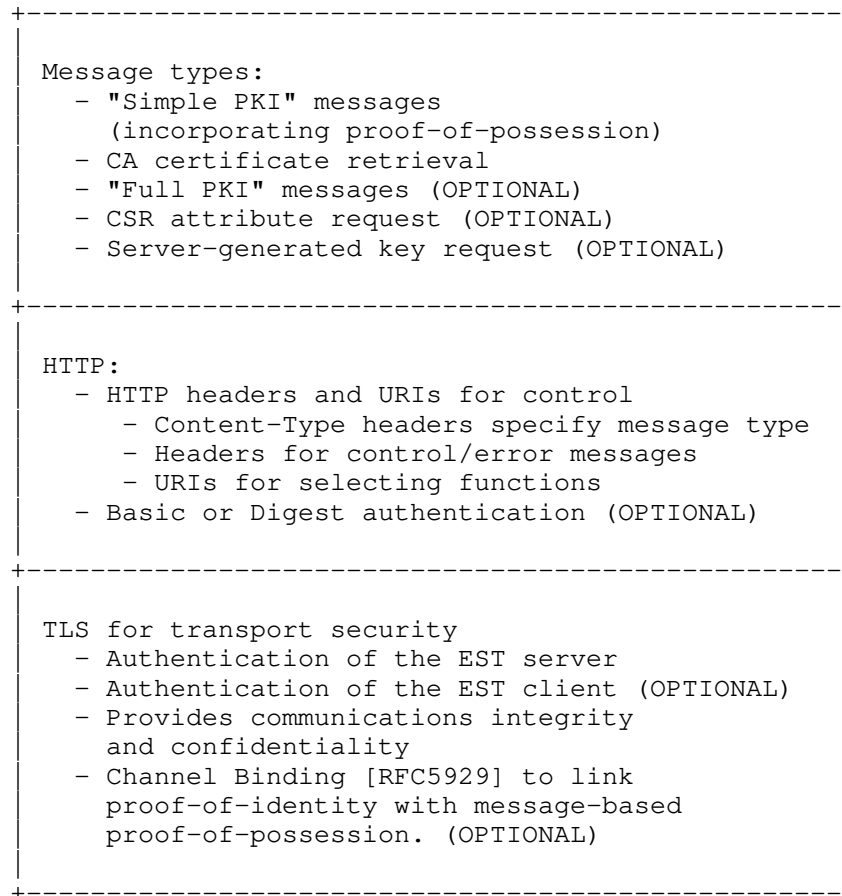


Figure 2

Specifying HTTPS as the secure transport for enrollment messages introduces two 'layers' to communicate authentication and control messages: TLS and HTTP.

The TLS layer provides integrity and confidentiality during transport. The proof-of-identity is supplied by TLS handshake authentication and optionally also by the HTTP layer headers. The message type and control/error messages are included in the HTTP headers.

CMC [RFC5272] Section 3.1 notes that "the Simple PKI Request MUST NOT

be used if a proof-of-identity needs to be included". Since the TLS and HTTP layers provide proof-of-identity for EST clients and servers the Simple PKI message types are used.

The TLS layer certificate exchange provides a method for authorizing client enrollment requests using existing certificates. Such certificates may have been issued by the CA (from which the client is requesting a certificate) or they may have been issued under a distinct PKI (e.g., an IEEE 802.1AR IDevID [IDevID] credential).

Proof-of-possession (PoP) is a distinct issue from proof-of-identity and is included in the Simple PKI message type as described in Section 3.4. A method of linking proof-of-identity and proof-of-possession is described in Section 3.5.

This document also defines transport for CMC [RFC5272] that complies with CMC Transport Protocols [RFC5273].

During protocol exchanges different certificates can be used. The following table provides an informative overview. End-entities MAY have one or more certificates of each type listed in Figure 3 and use one or more Trust Anchor databases of each type listed in Figure 4.

Certificates and their corresponding uses:

| Certificate            | Issuer   | Use and section references   |
|------------------------|--|--|
| EST server certificate | The CA served by the EST server                                      | Presented by the EST server during the TLS handshake<br><br>Section 3.3.1  |
| EST server certificate | A CA authenticatable by a third-party TA e.g., a web server CA       | Presented by the EST server during the TLS handshake<br><br>Section 3.3.1, and Security Considerations   |
| EST client certificate | A CA authenticatable by a third-party TA e.g., a device manufacturer | Presented by the EST client to the EST server by clients that have not yet enrolled<br><br>Section 3.3.2   |
| EST client certificate | The CA served by the EST server                                      | Presented by the EST client to PKI End Entities. Including to the EST server during future EST operations<br><br>Section 3.3.2                   |
| EST client certificate | The CA served by the EST server                                      | Presented by the EST client to PKI End Entities. Clients can obtain certs that cannot be used for EST client authentication<br><br>Section 4.2.1 |

Figure 3

Trust Anchor databases and their corresponding uses:

| TA database                              | Use and section references  |
|--|---|
| EST server<br>EST CA<br>TA database      | EST servers use this TA database to authenticate certificates issued by the EST CA, including EST client certificates during enroll/re-enroll operations<br><br>Section 3.3.2                     |
| EST server<br>Third-Party<br>TA database | EST servers use this TA database to authenticate certificates issued by third-party TAs.<br>e.g., EST client certificates issued by a device manufacturer<br><br>Section 3.3.2                    |
| EST client<br>Explicit<br>TA database    | EST clients use this TA database to authenticate certificates issued by the EST CA, including EST server certificates.<br><br>Section 3.1, Section 3.3.1, Section 3.6.1 and Section 4.1.1         |
| EST client<br>Implicit<br>TA database    | EST clients use this trust anchor database to authenticate an EST server that uses an externally issued certificate.<br><br>Section 3.1, Section 3.3.1, Section 3.6.2 and Security Considerations |

Figure 4

### 3.1. Application Layer

The EST client MUST be capable of generating and parsing Simple PKI messages (see Section 4.2). Generating and parsing Full PKI messages is OPTIONAL (see Section 4.3). The client MUST also be able to request CA certificates from the EST server and parse the returned "bag" of certificates (see Section 4.1). Requesting CSR attributes and parsing the returned list of attributes is OPTIONAL (see Section 4.5).

Details of the EST client application configuration are out of scope

of the protocol discussion but are necessary for understanding the prerequisites of initiating protocol operations. The EST client is RECOMMENDED to be configured with TA databases for Section 3.3.1 or with a secret key for Section 3.3.3. Implementations conforming to this standard MUST provide the ability to designate Explicit TAs. For human usability reasons a "fingerprint" of an Explicit TA database entry can be configured for bootstrapping as discussed in Section 4.1.1. Configuration of an Implicit TA database, perhaps by its inclusion within the EST client distribution or available from the operating system, provides flexibility along with the caveats detailed in Section 7. Implementations conforming to this standard MUST provide the ability to disable use of any Implicit TA database.

The EST client is configured with sufficient information to form the EST server URI. This can be the full operation path segment (e.g. `https://www.example.com/.well-known/est/` or `https://www.example.com/.well-known/est/arbitraryLabel1`) or the EST client can be configured with a tuple composed of the authority portion of the URI along with the OPTIONAL label (e.g. `"www.example.com:80"`, `"arbitraryLabel1"`) or just the authority portion of the URI.

### 3.2. HTTP Layer

HTTP is used to transfer EST messages. URIs are defined for handling each media type (i.e., message type) as described in Section 3.2.2. HTTP is also used for client authentication services when TLS client authentication is not available, due to lack of a client certificate suitable for use by TLS (see Section Section 3.2.3). HTTP authentication can also be used in addition to TLS client authentication if the EST server wishes additional authentication information, as noted in Section 2.2.3. Registered media types are used to convey EST messages as specified in Figure 6.

HTTP 1.1 [RFC2616] and above support persistent connections. As described in Section 8.1 of that RFC, persistent connections may be used to reduce network and processing load associated with multiple HTTP requests. EST does not require or preclude persistent HTTP connections and their use is out of scope of this specification.

#### 3.2.1. HTTP headers for control

This document profiles the HTTP content-type header (as defined in [RFC2046], but see Figure 6 for specific values) to indicate the media type for EST messages and to specify control messages for EST. The HTTP Status value is used to communicate success or failure of an EST function. HTTP authentication is used by a client when requested by the server.

The media types indicated in the HTTP content-type header indicates which EST message is being transferred. Media types used by EST are specified in Section 3.2.4.

### 3.2.2. HTTP URIs for control

The EST server MUST use the [RFC5785] defined path-prefix of `"/.well-known/"` and the registered name of `"est"`. Thus a valid EST server URI path begins with `"https://www.example.com/.well-known/est"`. Each EST operation is indicated by a path-suffix that indicates the intended operation:

Operations and their corresponding URIs:

| Operation                                | Operation Path  | Details       |
|--|-----------------|---------------|
| Distribution of CA certificates (MUST)   | /CACerts        | Section 4.1   |
| Enrollment of new clients (MUST)         | /simpleEnroll   | Section 4.2.  |
| Re-Enrollment of existing clients (MUST) | /simpleReEnroll | Section 4.2.2 |
| Full CMC (OPTIONAL)                      | /fullCMC        | Section 4.3   |
| Server-side Key Generation (OPTIONAL)    | /serverKeyGen   | Section 4.4   |
| Request CSR attributes (OPTIONAL)        | /CSRAttrs       | Section 4.5   |

Figure 5

The operation path (Figure 5) is appended to the path-prefix to form the URI used with HTTP GET or POST to perform the desired EST operation. An example valid URI absolute path for the `"/CACerts"` operation is `"/.well-known/est/CACerts"`. To retrieve the CA's certificates, the EST client would use the following HTTP request:

```
GET /.well-known/est/CACerts HTTP/1.1
```

Likewise, to request a new certificate in this example scheme, the EST client would use the following request:

```
POST /.well-known/est/simpleEnroll HTTP/1.1
```

The use of distinct operation paths simplifies implementation for servers that do not perform client authentication when distributing /CACerts responses.

An EST server MAY provide service for multiple CAs as indicated by an OPTIONAL additional path segment between the registered application name and the operation path. To avoid conflict the CA label MUST NOT be the same as any defined operation path segment. The EST server MUST provide services when the additional path segment is not included. The following are three example valid URIs:

1. `https://www.example.com/.well-known/est/CACerts`
2. `https://www.example.com/.well-known/est/arbitraryLabel1/CACerts`
3. `https://www.example.com/.well-known/est/arbitraryLabel2/CACerts`

In this specification the distinction between enroll and renew/rekey is explicitly indicated by the HTTP URI. When requesting /fullCMC operations CMC uses the same messages for certificate renewal and certificate rekey.

An EST server MAY provide additional services using other URIs.

### 3.2.3. HTTP-Based Client Authentication

The EST server MAY request HTTP-based client authentication. This request can be in addition to successful TLS client authentication (Section 3.3.2) if EST server policy requires additional authentication. (For example the EST server may require that an EST client "knows" a password in addition to "having" an existing client certificate). Or HTTP-based client authentication can be an EST server policy specified fallback in situations where the EST client did not successfully complete the TLS client authentication. (This might arise if the EST client is enrolling for the first time or if the certificates available to an EST client cannot be used for TLS client authentication).

HTTP Basic and Digest authentication MUST only be performed over TLS 1.1 [RFC4346] or later versions. As specified in CMC: Transport Protocols [RFC5273] the server "MUST NOT assume client support for any type of HTTP authentication such as cookies, Basic authentication, or Digest authentication". Clients SHOULD support the Basic and Digest authentication mechanism.

Servers that wish to use Basic and Digest authentication reject the HTTP request using the HTTP defined WWW-Authenticate response-header ([RFC2616], Section 14.47). The client is expected to retry the

request, including the appropriate Authorization Request Header ([RFC2617], Section 3.2.2), if the client is capable of using the Basic or Digest authentication. If the client is not capable of retrying the request or it is not capable of Basic or Digest authentication, then the client MUST terminate the connection.

A client MAY set the username to the empty string ("") if it is presenting a password that is not associated with a username.

Support for HTTP-based client authentication has security ramifications as discussed in Section 7. The client MUST NOT respond to the server's HTTP authentication request unless the client has authenticated the EST server (as per Section 3.6).

#### 3.2.4. Message types

This document uses existing media types for the messages as specified by [RFC2585], [RFC5967], and CMC [RFC5272]. To support distribution of multiple certificates for a CA certificate path, the [RFC2046] multipart/mixed media type is used.

For consistency with [RFC5273], each distinct EST message type uses an HTTP Content-Type header with a specific media type.

The EST messages and their corresponding media types are:

| Message type               | Request media type<br>Response media type(s)<br>Source(s) of types   | Request section(s)<br>Response section |
|----------------------------|--|--|
| CA certificate request     | N/A<br>application/pkcs7-mime<br>[RFC5751]   | Section 4.1<br>Section 4.1.1           |
| Cert enroll/renew/rekey    | application/pkcs10<br>application/pkcs7-mime<br>[RFC5967] [RFC5751]  | Section 4.2/4.2.1<br>Section 4.2.2     |
| Full CMC                   | application/pkcs7-mime<br>application/pkcs7-mime<br>[RFC5751]  | Section 4.3.1<br>Section 4.3.2         |
| Server-side Key Generation | application/pkcs10<br>multipart/mixed<br>(application/pkcs7-mime &<br>application/pkcs8)<br>[RFC5967] [RFC5751] &<br>[RFC5958] | Section 4.4.1<br>Section 4.4.2         |
| Request CSR attributes     | N/A<br>application/csrattrs<br>This RFC  | Section 4.5.1<br>Section 4.5.2         |

Figure 6

### 3.3. TLS Layer

TLS provides authentication, which in turn enables authorization decisions. The EST server and EST client are responsible for ensuring that an acceptable cipher suite is negotiated and that bidirectional authentication has been performed. Alternately, certificate-less TLS authentication, where neither the client nor server present a certificate, is also an acceptable method for EST mutual authentication.

HTTPS [RFC2818] specifies how HTTP messages are carried over TLS. HTTPS MUST be used. TLS 1.1 [RFC4346] (or a later version) MUST be used. TLS session resumption [RFC5077] SHOULD be supported.

TLS channel binding information MAY be inserted into a certificate request as detailed in Section 3.5 in order to provide the EST server

with assurance that the authenticated TLS client has access to the private key for the certificate being requested.

### 3.3.1. TLS-Based Server Authentication

The EST server MUST be authenticated during the TLS handshake unless the client is requesting Bootstrap Distribution of CA certificates (Section 4.1.1) or Full CMC (Section 4.3).

The EST client authenticates the EST server as defined for the cipher suite negotiated. The following text provides details assuming a certificate-based cipher suite, such as the TLS 1.1 [RFC4346] mandatory cipher suite (TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA). As an alternative to authentication using a certificate, an EST client MAY support certificate-less TLS authentication (Section 3.3.3).

Certificate validation MUST be performed as per [RFC5280]. The EST server certificate MUST conform to the [RFC5280] certificate profile.

The client validates the TLS server certificate using the EST client Explicit and, if enabled, Implicit TA database(s). The client MUST maintain a distinction between the use of Explicit and Implicit TA databases during authentication in order to support proper authorization. The EST client MUST perform authorization checks as specified in Section 3.6.

If certificate validation fails, the client MAY follow the procedure outlined in Section 4.1.1 for bootstrap distribution of CA certificates.

### 3.3.2. TLS-Based Client Authentication

TLS client authentication is the RECOMMENDED method for identifying EST clients. HTTP-Based Client Authentication (Section 3.2.3) MAY be used.

The EST server authenticates the EST client as defined for the cipher suite negotiated. The following text provides details assuming a certificate-based cipher suite such as the TLS 1.1 [RFC4346] mandatory cipher suite (TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA). The EST server MUST support certificate based client authentication. As an alternative or as an addition to authentication using a certificate, an EST server MAY support certificate-less TLS authentication (Section 3.3.3).

Generally, the client will use an existing certificate for renew or rekey operations. If the certificate to be renewed or rekeyed is appropriate for the negotiated cipher suite, then the client MUST use

it for the TLS handshake, otherwise the client SHOULD use an alternate certificate that is suitable for the cipher suite and contains the same subject identity information. When requesting an enroll operation the client MAY use a third-party issued client certificate to authenticate itself. The EST server MUST perform authorization checks as specified in Section 3.7.

If a client does not support TLS client authentication, then it MUST support HTTP-based client authentication (Section 3.2.3) or certificate-less TLS authentication (Section 3.3.3).

### 3.3.3. Certificate-less TLS Mutual Authentication

Certificate-less TLS cipher suites provide a way to perform mutual authentication in situations where neither the client nor server have certificates, do not desire to use certificates, or do not have the trust anchors necessary to verify a certificate. The client and server MAY negotiate a certificate-less cipher suite for mutual authentication.

When using certificate-less mutual authentication in TLS for enrollment, the cipher suite MUST be based on a protocol that is resistant to dictionary attack and MUST be based on a zero knowledge protocol. TLS-SRP ciphersuites listed in section 2.7 of [RFC5054] are suitable for this purpose. Section 7 lists the characteristics of a ciphersuite that are suitable for use in certificate-less mutual authentication for enrollment.

Successful authentication using a certificate-less cipher suite proves knowledge of a pre-shared secret which implicitly authorizes a peer in the exchange.

### 3.4. Proof-of-Possession

As defined in Section 2.1 of CMC [RFC5272], Proof-of-possession (POP) "refers to a value that can be used to prove that the private key corresponding to the public key is in the possession and can be used by an end-entity."

The signed enrollment request provides a signature-based proof-of-possession. The mechanism described in Section 3.5 strengthens this by optionally including "Direct"-based proof-of-possession [RFC5272] by including TLS session-specific information within the data covered by the enrollment request signature (thus linking the enrollment request to the authenticated end-point of the TLS connection).

### 3.5. Linking Identity and PoP information

Server policy will determine whether the server requires the mechanism specified in this section be used by the client. This specification provides an OPTIONAL method of linking identity and proof-of-possession by including information specific to the current authenticated TLS session within the signed certification request. The client can determine if the server requires the linking of identity and PoP by examining the CSR Attributes Response (see Section 4.5.2). Regardless of the CSR Attributes Response, clients are RECOMMENDED to link identity and PoP by embedding tls-unique information in the certification request. If tls-unique information is included by the client, the server MUST verify it. The EST server MAY reject requests without tls-unique information as indicated by server policy.

Linking identity and proof-of-possession proves to the server that the authenticated TLS client has possession of the private key associated with the certification request and that the client was able to sign the certification request after the TLS session was established. This is an alternative to the [RFC5272] Section 6.3-defined "Linking Identity and POP information" method available if Full PKI messages are used.

The client generating the request obtains the tls-unique value as defined in Channel Bindings for TLS [RFC5929] from the TLS subsystem. The tls-unique specification includes a synchronization problem as described in Channel Bindings for TLS [RFC5929] section 3.1. To avoid this problem, EST implementations that support this feature MUST use the tls-unique value from the first TLS handshake. EST clients and servers use their tls-unique implementation specific synchronization methods to obtain this first tls-unique value. TLS "secure\_renegotiation" [RFC5746] MUST be used. This maintains the binding from the first tls-unique value across renegotiations to the most recently negotiated connection.

The tls-unique value is base 64-encoded as specified in Section 4 of [RFC4648] and the resulting string is placed in the certification request challenge-password field ([RFC2985], Section 5.4.1). If tls-unique information is not embedded within the certification request the challenge-password field MUST be empty to indicate that the client did not include the optional channel-binding information (any value submitted is verified by the server as tls-unique information).

If the EST server makes use of a back-end infrastructure for processing, it is RECOMMENDED that the results of this verification be communicated. (For example this communication might use the CMC "RA POP Witness Control" in a CMC Full PKI Request message. Or an

EST server might TLS authenticate an EST client as being a trusted infrastructure element that does not forward invalid requests. A detailed discussion of back-end processing is out of scope).

When rejecting requests, the EST server response is as described for all enroll responses (Section 4.2.3). If a Full PKI Response is included, the CMCFailInfo MUST be set to popFailed. If a human readable reject message is included it SHOULD include an informative text message indicating that linking of identity and POP information is required.

### 3.6. Server Authorization

The client MUST check EST server authorization before accepting any server responses or responding to HTTP authentication requests.

The EST client authorization method depends on which method was used to authenticate the server. When the Explicit TA database is used to authenticate the EST server then Section 3.6.1 applies. When the Implicit TA database is used to authenticate the EST server then Section 3.6.2 applies. Successful authentication using a certificate-less cipher suite implies authorization of the server.

The client MAY perform bootstrapping as specified in Section 4.1.1 even if these checks fail.

#### 3.6.1. Client use of Explicit TA Database

When the EST client Explicit TA database is used to validate the EST server certificate the client MUST check either the configured URI against the server's identity, or the EST server certificate MUST contain the id-kp-cmcRA [RFC6402] extended key usage extension.

#### 3.6.2. Client use of Implicit TA Database

When the EST client Implicit TA database is used to validate the EST server certificate, the client MUST check the URI "against the server's identity as presented in the server's Certificate message" (HTTP Over TLS Section 3.1 "Server Identity" [RFC2818] and [RFC6125]). The provisioned URI provides the basis for authorization, and the server's authenticated identity confirms it is the authorized server.

### 3.7. Client Authorization

The decision to issue a certificate to a client is always controlled by local CA policy. The EST server configuration reflects this CA policy. This document does not specify any constraints on such

policy. EST provides the EST server access to each client's authenticated identity -- e.g., the TLS client's certificate in addition to any HTTP user authentication credentials -- to help in implementing such policy.

If the client's certificate was issued by the EST CA, and it includes the id-kp-cmcRA [RFC6402] extended key usage extension, then the client is a Registration Authority (RA) as described in [RFC5272] and [RFC6402]. In this case the EST server SHOULD apply authorization policy consistent with an RA client. For example when handling /simpleEnroll requests the EST server could be configured to accept PoP linking information that does not match the current TLS session because the authenticated EST client RA has verified this information when acting as an EST server (as specified in Section 3.5). More specific RA mechanisms are available if the EST client uses /fullCMC methods.

#### 4. Protocol Exchange Details

Before processing a request, an EST server determines if the client is authorized to receive the requested services. Likewise, the client determines if it will make requests to the EST server. These authorization decisions are described in the next two sections. Assuming that both sides of the exchange are authorized, then the actual operations are as described in subsequent sections.

##### 4.1. Distribution of CA certificates

The EST client can request a copy of the current CA certificates. This function is generally performed before other EST functions.

###### 4.1.1. Bootstrap Distribution of CA certificates

It is possible that the client was not configured with the TA database(s) necessary to validate the EST server certificate. This section describes a method by which minimally configured EST clients can populate their Explicit TA database.

If the EST client application does not specify either an Explicit TA database or a Implicit TA database then the initial TLS server authentication and authorization will fail. The client MAY provisionally continue the TLS handshake to completion for the purposes of accessing the /CACerts or /fullCMC method. If the EST client continues with an unauthenticated connection, the client MUST extract the HTTP content data from the response (Section 4.1.3 or Section 4.3.2) and engage a human user to authorize the CA certificate using out-of-band data such as a CA certificate

"fingerprint" (e.g., a SHA-256 or SHA-512 [SHS] hash on the whole CA certificate). In a /fullCMC response it is the Publish Trust Anchors control within the Full PKI Response that must be accepted manually. It is incumbent on the user to properly verify the TA information, or to provide the "fingerprint" data during configuration that is necessary to verify the TA information.

HTTP authentication requests MUST NOT be responded to if the server has not been authenticated.

The EST client uses the /CACerts response to establish an Explicit Trust Anchor database for subsequent TLS authentication of the EST server. EST clients MUST NOT engage in any other protocol exchange until after the /CACerts response has been accepted and a new TLS session has been established (using TLS certificate-based authentication).

#### 4.1.2. Distribution of CA certificates request

EST clients request the Explicit TA database information of the CA (in the form of certificates) with an HTTPS GET message using an operation path of "/CACerts". EST clients and servers MUST support the /CACerts function. Clients SHOULD request an up-to-date response before stored information has expired in order to ensure the EST CA TA database is up to date.

The EST server MUST NOT require client authentication or authorization to reply to this request.

The client MUST authenticate the EST server as specified in Section 3.3.1 and Section 3.3.3 and check the server's authorization as given in Section 3.6 or follow the procedure outlined in Section 4.1.1.

#### 4.1.3. Distribution of CA certificates response

The EST server responds to a Distribution of CA certificates request with the EST CA certificates within a Simple PKI Response. If the certificates are successfully returned, the server response MUST have an HTTP 200 response code with a content-type of "application/pkcs7-mime". Any other response code indicates an error and the client MUST abort the protocol.

The EST server MUST include the current root CA certificate in the response. The EST server MUST include any additional certificates the client would need to build a chain from an EST CA issued certificate to the current EST CA TA. For example if the EST CA is a subordinate CA then all the appropriate subordinate CA certificates

necessary to build a chain to the root EST CA are included in the response.

The EST server SHOULD include the three "Root CA Key Update" certificates OldWithOld, OldWithNew, and NewWithOld in the response chain. These are defined in Section 4.4 of CMP [RFC4210]. The EST client MUST be able to handle these certificates in the response. The EST CA's most recent self-signed certificate (e.g. NewWithNew certificate) is self-signed and has the latest NotAfter date. If the EST server does not include these in the response then after the current EST CA certificate expires the EST clients will need to be (re)enrolled with the PKI using the Bootstrap Distribution of CA certificates (Section 4.1.1) method which involves user interaction.

If the CA certificate was authorized in an out-of-band manner then, after such validation occurs, all the other certificates MUST be validated using normal [RFC5280] certificate path validation (using the most recent CA certificate as the TA) before they can be used to build certificate paths during certificate validation.

The EST client MUST store the extracted EST CA certificate as an Explicit TA database entry for subsequent EST server authentication. The EST client SHOULD disable use of Implicit TA database entries for this EST server, now that an Explicit TA database entry is available. If the client does this then the client MUST include a "Trusted CA Indication" extension in future TLS sessions [RFC4366] to indicate to the server that only an EST Server certificate authenticatable by the Explicit TA database entry is acceptable. The EST client also makes the CA Certificate response information available to the end-entity software for use when validating peer certificates.

The response format is the CMC Simple PKI Response, as defined in [RFC5272]. The HTTP content-type of "application/pkcs7-mime" is used. The Simple PKI response is base 64-encoded, as specified in Section 4 of [RFC4648], and sandwiched between headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBFBMqswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of base 64 encoding of CMC Simple PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVVeryble4iZgMUvuIgwEjQwpD
8J4OhHvLhlo=
-----END PKCS7-----
```

#### 4.2. Client Certificate Request Functions

EST clients request a certificate from the EST server with an HTTPS POST using the operation path value of "/simpleEnroll". EST clients request a renew/rekey of existing certificates with an HTTP POST

using the operation path value of `"/simpleReEnroll"`. EST servers MUST support the `/simpleEnroll` and `/simpleReEnroll` functions.

It is RECOMMENDED that a client obtain the current CA certificates, as described in Section 4.1, before performing certificate request functions. This ensures that the client will be able to validate the EST server certificate. The client MUST authenticate the EST server as specified in Section 3.3.1 and Section 3.3.3. The client MUST verify the authorization the EST server as specified in Section 3.6.

The server MUST authenticate the client as specified in Section 3.3.2 and Section 3.3.3. The server MUST verify client authorization as specified in Section 3.7. The EST server MUST check the `tls-unique` value as described in Section 3.5 if one is submitted by the client.

The server MAY accept a certificate request for manual authorization checking by an administrator. (Section 4.2.3 describes the use of an HTTP 202 response to the EST client if this occurs).

#### 4.2.1. Simple Enrollment of Clients

When HTTPS POSTing to `/simpleEnroll` the client MUST include a Simple PKI Request as specified in CMC Section 3.1 (i.e., a PKCS#10 Certification Request).

The Certification Signing Request (CSR) signature provides proof-of-possession of the private key to the EST server. If the CSR `KeyUsage` extension indicates the private key can be used to generate digital signatures then the CSR signature MUST be generated using the private key. If the key can be used to generate digital signatures but the requested CSR `KeyUsage` extension prohibits generation of digital signatures then the CSR signature MUST still be generated using the private key but the key MUST NOT be used to for any other signature operations (this is consistent with the recommendations concerning submission of proof-of-possession to an RA or CA as described in [SP-800-57-Part-1]). The use of `/fullCMC` operations provides access to more advanced proof-of-possession methods that MUST be used when the key pair can not be used for digital signature generation (see Section 4.3).

The HTTP content-type of `"application/pkcs10"` is used here. The format of the message is as specified in Section 6.4 of [RFC4945].

The EST client MAY request additional certificates even when using an existing certificate in the TLS client authentication. For example the client can use an existing certificate for TLS client authentication when requesting a certificate that cannot be used for TLS client authentication.

#### 4.2.2. Simple Re-Enrollment of Clients

EST clients renew/rekey certificates with an HTTPS POST using the operation path value of `"/simpleReEnroll"`.

A certificate request employs the same format as the `"simpleEnroll"` request, using the same HTTP content-type. The request Subject field and SubjectAltName extension MUST be identical to the corresponding fields in the certificate being renewed/rekeyed. The ChangeSubjectName attribute, as defined in [RFC6402], MAY be included in the CSR to request that these fields be changed in the new certificate.

If the Subject Public Key Info in the certification request is the same as the current client certificate, the EST server performs a renew operation. If the public key information is different than the currently issued certificate then the EST server performs a rekey operation.

#### 4.2.3. Simple Enroll and Re-Enroll Response

If the enrollment is successful, the server response MUST contain an HTTP 200 response code with a content-type of `"application/pkcs7-mime"`. The response data is a certs-only Simple PKI Response containing only the certificate that was issued. The Simple PKI response is base 64-encoded and sandwiched between headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of base 64 encoding of CMC Simple PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVVeryble4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PKCS7-----
```

When rejecting a request the server MUST specify either an HTTP 4xx error, or an HTTP 5xx error. A Simple PKI Response with an HTTP content-type of `"application/pkcs7-mime"` (see Section 4.3.2) MAY be included in the response data to convey an error response. If the content-type is not set the response data MUST be a plain text human-readable error message containing informative information describing why the request was rejected (for example indicating that CSR attributes are incomplete).

If the server responds with an HTTP [RFC2616] 202, this indicates that the request has been accepted for processing but that a response is not yet available. The server MUST include a Retry-After header as defined for HTTP 503 responses. The server also MAY include informative human-readable content. The client MUST wait at least

the specified 'retry-after' time before repeating the same request. The client repeats the initial enrollment request after the appropriate 'retry-after' interval has expired. The client SHOULD log or inform the end user of this event. The server is responsible for maintaining all state necessary to recognize and handle retry operations as the client is stateless in this regard; it simply sends the same request repeatedly until it receives a different response code.

All other return codes are handled as specified in HTTP [RFC2616].

#### 4.3. Full CMC

An EST client can request a certificate from an EST server with an HTTPS POST using the operation path value of "/fullCMC". Support for the /fullCMC function is OPTIONAL for both clients and servers.

##### 4.3.1. Full CMC Request

If the HTTP POST to /fullCMC is not a valid Full PKI Request, the server MUST reject the message. The HTTP content-type used is "application/pkcs7-mime", as specified in [RFC5273].

##### 4.3.2. Full CMC Response

The server responds with the client's newly issued certificate or provides an error response.

If the enrollment is successful, the server response MUST include an HTTP 200 response code with a content-type of "application/pkcs7-mime" as specified in [RFC5273]. The response data includes either the Simple PKI Response or the Full PKI Response as specified in Section 3.2 of [RFC5272].

When rejecting a request, the server MUST specify either an HTTP 4xx error or an HTTP 5xx error. A CMC response with content-type of "application/pkcs7-mime" SHOULD be included in the response data for any CMC error response. If the content-type is not set the response data MUST be a plain text human-readable error message containing informative information describing why the request was rejected (for example indicating that CSR attributes are incomplete).

All other return codes are handled as specified in Section 4.2.3 or HTTP [RFC2616]. For example, a client interprets an HTTP 404 or 501 response to indicate that this service is not implemented.

The Full PKI Response is base 64-encoded and sandwiched between headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of base 64 encoding of CMC Full PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVVeryble4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PKCS7-----
```

#### 4.4. Server-side Key Generation

An EST client may request a private key and associated certificate from an EST server using an HTTPS POST with an operation path value of `"/serverKeyGen"`. Support for the `/serverKeyGen` function is OPTIONAL.

A client MUST authenticate an EST server as specified in Section 3.3.1 and Section 3.3.3.

The EST server MUST authenticate the client as specified in Section 3.3.2 and Section 3.3.3. The server SHOULD use Client Authorization for authorization purposes. The EST server applies whatever authorization or logic it chooses to determine if the private key and certificate should be provided.

Proper random number and key generation [RFC4086] is a server implementation responsibility and server storage of generated keys is a local option. The key pair and certificate are transferred over the TLS session. The cipher suite used to return the private key and certificate MUST offer confidentiality commensurate with the private key being delivered to the client.

The EST client MAY request additional certificates even when using an existing certificate in the TLS client authentication. For example the client can use an existing certificate for TLS client authentication when requesting a certificate that cannot be used for TLS client authentication.

##### 4.4.1. Server-side Key Generation Request

The certificate request is HTTPS POSTed and is the same format as for the `"/simpleEnroll"` and `"/simpeReEnroll"` path extensions with the same content-type.

In all respects the server SHOULD treat the CSR as it would any enroll or re-enroll CSR; the only distinction here is that the server MUST ignore the public key values and signature in the CSR. These are included in the request only to allow re-use of existing codebases for generating and parsing such requests.

If the client desires to receive the private key with encryption that exists outside and in addition to that of the TLS transport used by EST or if server policy requires that the key be delivered in such a form, the client MUST include a DecryptKeyIdentifier attribute (as defined in Section 2.2.5, [RFC4108]) specifying the identifier of the secret key to be used by the server to encrypt the private key. While that attribute was originally designated for specifying a firmware encryption key, it exactly mirrors the requirements for specifying a private key encryption key. If the server does not have a secret key matching the identifier specified by the client, the request must be terminated and an error returned to the client. Distribution of the key specified by the DecryptKeyIdentifier to the key generator and the client is outside the scope of this document.

#### 4.4.2. Server-side Key Generation Response

If the request is successful, the server response MUST have an HTTP 200 response code with a content-type of "multipart/mixed" consisting of two parts. One part is the private key data and the other part is the certificate data.

The format in which the private key data part is returned is dependent on whether the private key is being returned with additional encryption on top of that provided by TLS.

- o If additional encryption is being employed, the private key is placed inside of a CMS SignedData. The SignedData is signed by the party that generated the private key, which may or may not be the EST server. The SignedData is further protected inside of a CMS EnvelopedData, as described in Section 4 of [RFC5958]. The EnvelopedData content is encrypted using the secret key identified in the request. The EnvelopedData is returned in the response as an "application/pkcs7-mime" part.
- o If additional encryption is not being employed, the private key data MUST be an "application/pkcs8". An "application/pkcs8" part consists of the base 64-encoded DER-encoded PrivateKeyInfo sandwiched between headers as described in [RFC5958].

The certificate data part is an "application/pkcs7-mime" and exactly matches the certificate response to /simpleEnroll. If both parts are "application/pkcs7-mime" the client checks each; one will be a certs-only Simple PKI response and the other will be the CMS message with the encrypted data.

```
-----BEGIN PRIVATE KEY-----
MIIBhDCB7gIBADBQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of base 64 encoding of DER-encoded PrivateKeyInfo
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVVeryble4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PRIVATE KEY-----
```

or

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of CMS secured Private key
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVVeryble4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PKCS7-----
```

When rejecting a request, the server MUST specify either an HTTP 4xx error, or an HTTP 5xx error. If the content-type is not set, the response data MUST be a plain text human-readable error message.

#### 4.5. CSR Attributes

CA policy may allow inclusion of client-provided attributes in certificates that it issues, and some of these attributes may describe information that is not available to the CA. In addition, a CA may desire to certify a certain type of public key and a client may not have a priori knowledge of that fact. Therefore, clients SHOULD request a list of expected attributes that are required, or desired, by the CA in an enrollment request, or if dictated by local policy.

Requesting CSR Attributes is optional but clients are advised that CA's may refuse enrollment requests that are not encoded according to the CA's policy.

##### 4.5.1. CSR Attributes Request

The EST client requests a list of CA-desired CSR attributes from the CA by sending an HTTPS GET message to the EST server with an operations path of `"/CSRAttrs"`.

##### 4.5.2. CSR Attributes Response

If locally configured policy for an authenticated EST client indicates a CSR Attributes Response is to be provided, the server response MUST include an HTTP 200 response code. An HTTP response code of 204 or 404 indicates that a CSR Attributes Response is not available. Regardless of the response code, the EST server and CA

MAY reject any subsequent enrollment requests for any reason, e.g., incomplete CSR attributes in the request.

If the CA requires a particular crypto system (e.g., certification of a public key based on a certain elliptic curve) it MUST provide that information in the CSR Attributes response. If an EST server requires the linking of identity and PoP information (see Section 3.5) it MUST include the challengePassword OID in the CSR Attributes response.

Responses to attribute request messages MUST be encoded as content type "application/csrattrs". The syntax for application/csrattrs body is as follows:

Csrattrs ::= SEQUENCE SIZE (0..MAX) OF OBJECT IDENTIFIER { }

An EST server includes zero or more object identifiers that it requests the client to include in a certification request. When the server encodes Csrattrs as an empty SEQUENCE it means that the server has no specific additional attributes it requests in a client certification request (this is functionally equivalent to an HTTP response code of 204 or 404.) The sequence is Distinguished Encoding Rules (DER) encoded and then base 64 encoded (section 4 of [RFC4648]). The resulting text forms the application/csrattr body, without headers.

For example, if a CA requests a client to submit a certification request containing the Media Access Control (MAC) address [RFC2397] of a device, the challengePassword (indicating that Linking of Identity and POP information is requested, see Section 3.5), to use the the secp384r1 elliptic curve, and to use the SH384 hash function then it sends the following object identifiers:

- o macAddress: 1.3.6.1.1.1.1.22
- o challengePassword: 1.2.840.113549.1.9.7
- o the secp384r1 elliptic curve: 1.3.132.0.4
- o the SHA384 hash function: 2.16.840.1.101.3.4.2.2

and encodes them into an ASN.1 SEQUENCE to produce:

```
30 26 06 07 2B 06 01 01 01 01 16 06 09 2A 86 48 86 F7 0D 01 09 07
06 05 2B 81 04 00 04 06 09 60 86 48 01 65 03 04 02 02
```

and then base 64 encodes the resulting ASN.1 SEQUENCE to produce:

MCYGBysGAQEBARYGCSqGSib3DQEJBwYFK4EEAAQGCWCGSAFlAwQCAg==

The EST client parses the OID's in the response and handles each OID independently. When an OID indicates a known descriptive CSR attribute type, the client SHOULD include the requested information in the subsequent CSR that it submits, either in the CSR attributes or in any other appropriate CSR field. When an OID indicates a particular way to generate the CSR, the client SHOULD generate its CSR according to the parsed OID. When an OID is of an unknown type the OID MUST be ignored by the client.

## 5. Contributors/Acknowledgements

The editors would like to thank Stephen Kent, Vinod Arjun, Jan Vilhuber, Sean Turner, Russ Housley, and others for their feedback and prototypes of early drafts. Our thanks also go to the authors of [RFC6403] around whose document we structured part of this specification.

## 6. IANA Considerations

IANA is requested to register the following:

IANA SHALL update the well-known URI registry with the following filled-in template from [RFC5785].

URI suffix: est

Change controller: IETF

IANA SHALL update the Application Media Types registry with the following filled-in template from [RFC4288].

The media subtype for Attributes in a CertificationRequest is application/csrattrs.

Type name: application

Subtype name: csrattrs

Required parameters: None

Optional parameters: None

Encoding considerations: binary;

**Security Considerations:**

Clients request a list of attributes that servers wish to be in certification requests. The request/response SHOULD be done in a TLS-protected tunnel.

Interoperability considerations: None

Published specification: This memo.

Applications which use this media type:

Enrollment over Secure Transport (EST)

Additional information:

Magic number(s): None

File extension: None

Macintosh File Type Code(s):

Person & email address to contact for further information:

Dan Harkins <dharkins@arubanetworks.com>

Restrictions on usage: None

Author: Dan Harkins <dharkins@arubanetworks.com>

Intended usage: COMMON

Change controller: The IESG

**7. Security Considerations**

Support for Basic authentication as specified in HTTP [RFC2617] allows the server access to a client's cleartext password. This provides support for legacy username/password databases but requires exposing the plaintext password to the EST server. Use of a PIN or one-time-password can help mitigate such exposure, but it is RECOMMENDED that EST clients use such credentials only once to obtain a client certificate (that will be used during future interactions with the EST server).

When a client uses the Implicit TA database for certificate validation (see Section 3) then authorization proceeds as specified

in Section 3.6.2. In this situation, the client has validated the server as being a certified-by-a-third-party responder for the URI configured, but cannot verify that the responder is authorized to act as an RA for the PKI in which the client is trying to enroll. Clients using an implicit trust anchor database are RECOMMENDED to only use TLS-based client authentication (to prevent exposing HTTP-based Client Authentication information). It is RECOMMENDED that such clients include "Linking Identity and POP information" (Section 3.5) in requests (to prevent such requests from being forwarded to a real EST server by a MITM). It is RECOMMENDED that the implicit trust anchor database used for EST server authentication be carefully managed, to reduce the chance of a third-party CA with poor certification practices from being trusted. Disabling the implicit trust anchor database after successfully receiving the Distribution of CA Certificates response (Section 4.1.3) limits any vulnerability to the first TLS exchange.

Certificate-less TLS ciphersuites that maintain security and perform the mutual authentication necessary for enrollment have the following properties:

- o the only information leaked by an active attack is whether a single guess of the secret is correct or not.
- o any advantage an adversary gains is through interaction and not computation.
- o it is possible to perform countermeasures, such as exponential backoff after a certain number of failed attempts, to frustrate repeated active attacks.

Using a certificate-less ciphersuite that does not have the properties listed above would render the results of enrollment void and potentially result in certificates being issued to unauthenticated and/or unauthorized entities.

When using a certificate-less TLS cipher suite, the shared secret used for authentication and authorization cannot be shared with an entity that is not a party to the exchange: someone other than the client and the server. Any additional sharing of secrets voids the security afforded by a certificate-less cipher suite. Exposure of a shared secret used by a certificate-less cipher suite to a third-party enables client impersonation that can result in corruption of a client's trust anchor database.

As described in CMC Section 6.7, "For keys that can be used as signature keys, signing the certification request with the private key serves as a POP on that key pair". The inclusion of tls-unique

within the certification request links the proof-of-possession to the TLS proof-of-identity by enforcing that the POP operation occurred while the TLS session was active. This implies to the server that the authenticated client currently has access to the private key. If the authenticated client is known to have specific capabilities, such as hardware protection for authentication credentials and key storage, this implication is strengthened but not proven.

The server-side key generation method allows keys to be transported over the TLS connection to the client. The distribution of private key material is inherently risky. Private key distribution uses the encryption mode of the negotiated TLS cipher suite. Keys are not protected by preferred key wrapping methods such as AES Key Wrap [RFC3394] or as specified in [RFC5958] as encryption of the private key beyond that provided by TLS is optional. It is RECOMMEND that EST servers not support this operation by default. It is RECOMMENDED that clients not request this service unless there is a compelling operational benefit. Use of an implicit trust anchor database is NOT RECOMMENDED when server-side key generation is employed. The use of an encrypted CMS Server-side Key Generation Response is RECOMMENDED.

Regarding the CSR attributes that the CA may list for inclusion in an enrollment request, there are no real inherent security issues with the content being conveyed but an adversary who is able to interpose herself into the conversation could exclude attributes that a server may want, include attributes that a server may not want, and render meaningless other attributes that a server may want.

## 8. References

### 8.1. Normative References

- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
- [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.
- [RFC2314] Kaliski, B., "PKCS #10: Certification Request Syntax Version 1.5", RFC 2314, March 1998.
- [RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP",

RFC 2585, May 1999.

- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC2617] 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.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, November 2000.
- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, August 2005.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005.
- [RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", RFC 4288, December 2005.
- [RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.
- [RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

- [RFC4945] Korver, B., "The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX", RFC 4945, August 2007.
- [RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, November 2007.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.
- [RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", RFC 5273, June 2008.
- [RFC5274] Schaad, J. and M. Myers, "Certificate Management Messages over CMS (CMC): Compliance Requirements", RFC 5274, June 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.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.
- [RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, February 2010.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.
- [RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, July 2010.
- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, August 2010.
- [RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, August 2010.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity

within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

- [RFC6402] Schaad, J., "Certificate Management over CMS (CMC) Updates", RFC 6402, November 2011.
- [SHS] National Institute of Standards and Technology, "Federal Information Processing Standard Publication 180-4: Secure Hash Standard (SHS)", March 2012, <<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>>.
- [X.680] ITU-T Recommendation, "ITU-T Recommendation X.680 Abstract Syntax Notation One (ASN.1): Specification of basic notation", November 2008, <<http://www.itu.int/rec/T-REC-X.680-200811-I/en>>.
- [X.690] ITU-T Recommendation, "ITU-T Recommendation X.690 ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", November 2008, <<http://www.itu.int/rec/T-REC-X.690-200811-I/en>>.

## 8.2. Informative References

- [IDevID] IEEE Std, "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, August 1998.
- [RFC2712] Medvinsky, A. and M. Hur, "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)", RFC 2712, October 1999.
- [RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.
- [RFC6403] Ziegler, L., Turner, S., and M. Peck, "Suite B Profile of Certificate Management over CMS", RFC 6403, November 2011.
- [SP-800-57-Part-1] National Institute of Standards and Technology, "Recommendation for Key Management - Part 1: General (Revision 3)", July 2012, <[http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57\\_part1\\_rev3\\_general.pdf](http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf)>.

[X.520] ITU-T Recommendation, "ITU-T Recommendation X.520 The Directory: Selected attribute types", November 2008, <<http://www.itu.int/rec/T-REC-X.520-200811-I/en>>.

## Appendix A. Operational Scenario Example Messages

(informative)

This section expands on the Operational Scenario Overviews by providing detailed examples of the messages at each TLS layer.

### A.1. Obtaining CA Certificates

The following is an example of a valid /CACerts exchange.

During the initial TLS handshake the client can ignore the optional server generated "certificate request" and can instead proceed with the HTTP GET request:

```
GET /CACerts HTTP/1.1
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenS
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
```

In response the server provides the current CA certificate:

```

<= Recv header, 38 bytes (0x26)
Content-Type: application/pkcs7-mime
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

<= Recv data, 1111 bytes (0x457)
-----BEGIN PKCS7-----.MIIDEQYJKoZIhvcNAQcCoIIDAjCCAv4CAQExADALBg
kqhkiG9w0BBwGggLkMIIC.4DCCAcigAwIBAgIJA0jxMZcXhE5wMA0GCSqGSIb3D
QEBBQUAMBCxFTATBgNVBAMT.DGVzdEV4YW1wbGVdQTAeFw0xMjA3MDQxODM5Mjda
Fw0xMzA3MDQxODM5MjdaMBcx.FTATBgNVBAMTDGVzdEV4YW1wbGVdQTCASiWdQY
JKoZIhvcNAQEBBQADggEPADCC.AQoCggEBALQ7SjZSt6qrnBzUnBNj9z4oxYkvMA
Vh0OIOVRkNhZ/2kDGsds0ne7cw.W33kYlXPba4psdLMixCT/O8ZQMpgA+QFKtwb9
VPE8EFUgGzxSYHQHjhJsbg0BVaN.Ya38vjKMjvosuSXUhwkvU57SInSkMr3/aNtS
T8qFfeC6Vuf/G/GLHGuHQKAY/DSO.206MjaMNmWYRVQQVERGookRA4GBF/YE+G/C
SlTsCQNE0KyBFz8JWIkgyY2gYkxb7.wWMvvhaU/Esp+2DG92v9Dhs2MRgrR+WPs7
Y6CYOLD5Mr5lEdkHg27IxkSAoRrI6D.fnVVEQGCj7QrrsUgfXFVYv6cCWFfhMcCA
wEAAAMvMC0wDAYDVR0TBAUwAwEB/zAd.BgNVHQ4EFgQUhH9KxW5Ts jkgL7kg2kxJ
yy5tD/MwDQYJKoZIhvcNAQEFBQADggEB.AD+vydZo292XFb2vXojdKD57Gv4tKVm
hvXRdVINntzkY/0AyFCfHJ4BwndgtMh4t.rvBD8+8dL+W3jFPjcSCcUQ/JEnFuMn
b5+kivLeqOnUshETasFPBz2Xq4C1sHDno9.CW0csjPPw08Tn4dSrZDBSsq1NDXB2z
9NOpaVnbp01qQGhXS0aEvcbZcDuGiW7Di3.gV++remokuPph/s6Xozffzc7ZVzf
Job6tS4RwNz01sutPybXiRWivOz7+QeCOT87.nTGlkQH/+RImUyJ2jefjAW/GDFT
Pzek6cZnabAtsg32n0Pv0j0/1RTNSdyGxPIVA.2f9fhMqMz+vm3w4CFNkGZnOhAD
EA.-----END PKCS7-----.

```

## A.2. Certificate TLS authentication

The following is an example of a valid /simpleEnroll exchange. During this exchange the EST client uses an existing certificate issued by a third-party CA to obtain an initial certificate from the EST server.

During the initial TLS handshake the server generated "certificate request" includes both the distinguished name of the EST CA ("estExampleCA") and it includes the distinguished name of a third-party CA ("estEXTERNALCA"):

```

0d 00 00 3d 03 01 02 40 00 37 00 1a 30 18 31 16 ...=...@.7..0.1.
30 14 06 03 55 04 03 13 0d 65 73 74 45 58 54 45 0...U....estEXTE
52 4e 41 4c 43 41 00 19 30 17 31 15 30 13 06 03 RNALCA..0.1.0...
55 04 03 13 0c 65 73 74 45 78 61 6d 70 6c 65 43 U....estExampleC
41                                     A

```

Which decodes as:

```

Acceptable client certificate CA names
/CN=estEXTERNALCA
/CN=estExampleCA

```

The EST client provides a certificate issued by "estEXTERNALCA" in the certificate response and the TLS handshake proceeds to completion. The EST server accepts the EST client certificate for authentication and accepts the EST client's POSTed certificate request:

```
POST /simpleEnroll HTTP/1.1
```

```
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenS  
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
```

```
Host: 127.0.0.1:8085
```

```
Accept: */*
```

```
Content-Type: application/pkcs10
```

```
Content-Length: 952
```

```
=> Send data, 952 bytes (0x3b8)
```

```
-----BEGIN CERTIFICATE REQUEST-----.MIICHjCCAW4CAQAwQTElMCMGA1UE  
AxMccmVxIGJ5IGNsaWVudCBpbjBkZW1vIHNO.ZXAgNjEYMBYGA1UEBRMPUE1E0ld  
pZGdl dCBTTjo2MIIIBIjANBgkqhkiG9w0BAQEF.AAOCAQ8AMIIBCgKCAQEawhYyI+  
aYezyx+kW0GVUBMKLf2BUd8BgGykkIJYxms6SH.Bv5S4ktcpYbEpR9iCmp96vK6a  
Ar57ArZtMmi0Y6eLX4c+njJnYhUeTivnfyfMM5d.hNVwyzKbJagm5f+RLTMfp0y0  
ykqrfZlhFhcNrRzF6mJeaORTHBehMdu8RXcbmy5R.s+vjnUC4Fe3/oLHtXePyYv1  
qqLkk0XDrw/+lx0y4Px5tiyb84iPnQOXjG2tuStM+.iEvfpNAnwU0+3GDj13sjx0  
+gTKvblp6Diw9NSaqIAKupcgWsA0JlyYkgPiJnXFKL.vy6rXoOyx3wAbGKLrKCxT  
l+RH3oNXf3UCH70aD758QIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBADwpafWU  
BsOJ2g2oyHQ7Ksw6MwvimjhB7GhjweCceTSLInUMk10.4E0TfNqaWcoQengMVZr  
IcbOb+sa69BWNb/WYIULfEtJIV23/g3n/y3JltMNw/q+R.200t0bNAViihQHmlF  
6dt93tkRrTzXnhV70Ijnff08G7P9HfnXQH4Eiv3zOB6Pak.JoL7QlWQ+w5vHpPo6  
WGH5n2iE+Ql76F0HykGegar402+ae0WlGLEvcN9wiFQVKh.KUHteU10SEPi j l q f  
QW+hciLleX2CwuZY5MqKb4qqyDTs4HSQCBC18jR2cXsGDuN4.PcMPp+9A1/UPuGD  
jhwPt/K3y6aV8zUEh8Ws=-----END CERTIFICATE REQUEST-----.
```

The EST server uses the trusted third party CA issued certificate to perform additional authorization and issues a certificate to the client:

```

<= Recv header, 38 bytes (0x26)
Content-Type: application/pkcs7-mime
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

<= Recv data, 1200 bytes (0x4b0)
-----BEGIN PKCS7-----.MIIDUQYJKoZIhvcNAQcCoIIDQjCCAz4CAQEExADALBg
kqhkiG9w0BBwGgggMkMIID.IDCCAgigAwIBAgIBBjANBgkqhkiG9w0BAQUFADAXM
RUwEwYDVQQDEwxc3RFeGft.cGxlQ0EwHhcNMTIwNzA0MTgzOTM3WhcNMTMwNzA0
MTgzOTM3WjBBMSUwIwYDVQQD.ExxyZXEGYnkgY2xpZW50IGluIGRlbW8gc3RlcCA
2MRgwFgYDVQQFEw9QSUQ6V2lk.Z2V0IFNOOjYwggEiMA0GCSqGSIb3DQEBAQUAA4
IBDwAwggEKAoIBAQDCFjIj5ph7.PLH6RbQZVRswot/YFR3wGAbKSQgljGazpIcG/
lLiSlylhsSlH2IKan3q8rpoCvns.Ctm0yaLRjp4tfhz6eMmdiFR5OK+d/J8wz12E
1XDLmpslqCbl/5EtMx+nTLTKSqt9.nWEWfw2tHMXqYl5o5FMcf6Ex27xFdxubLlG
z6+OdQLgV7f+gse1d4/Ji/WqqWSTR.cOvD/6XHTLg/Hm2LJvziI+dA5eMba25K0z
6IS9+k0CfBTT7cYOOXeyPHT6BMq9uW.noOLD01JqogAq6lyBawDQmXJiSA+ImdcU
ou/Lqteg7LHfABsYousoLFOX5Efegld./dQIfvRoPvnxAgMBAAGjTTBLMAkGA1Ud
EwQCMAAwHQYDVR0OBBYEFJv4oLLeNxNK.OMmQDDu jyNR+zaVPMB8GA1UdIwQYMBa
AFIR/SsVuU7I5IC+5INpMScsubQ/zMA0G.CSqGSIb3DQEBAQUAA4IBAQCm domfdR
9vi4VUYdF+eym7F8qVUG/1jtjfaxmrzKeZ.7LQ1F758RtwG9CDu2GPHNPjjeM+DJ
RQZN999eLs3Qd/DIJCNimaqdDqmkeBFC5hq.LZOxbKhSmhlr7YKjIZuyI299rOaI
W54ULyz8k0zw6R1/0lMJTsDFGJM+9yDeaARE.n3vtKnUDGHsVU3fYpDENaQUunoU
MZfuEdejfHhU7lVbJI1oSJbnRwBFkPr/RQ3/5.FymcrBD9RpAM5MsQIn0BONil/o
JM+LjOJqyZLbBxz6P3w/OiJGYJNfFT8YudLfjZ.LDX8A8FFcReapNELC4QxE4OrA
hN3sQUT2O7ndIsit4kJoQAxA==.-----END PKCS7-----.
```

### A.3. Username/Password Distributed Out-of-Band

The following is an example of a valid /simpleEnroll exchange. During this exchange the EST client uses an out-of-band distributed username/password to authenticate itself to the EST server.

During the initial TLS handshake the client can ignore the optional server generated "certificate request" and can instead proceed with the HTTP POST request:

```
POST /simpleEnroll HTTP/1.1
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenSSL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/pkcs10
Content-Length: 952
```

```
=> Send data, 952 bytes (0x3b8)
-----BEGIN CERTIFICATE REQUEST-----.MIICHjCCAW4CAQAwQTElMCMGA1UE
AxMccmVxIGJ5IGNsaWVudCBpbjBkZW1vIHNO.ZXAgMjEYMBYGA1UEBRMPUElEold
pZGldCBTTjoyMIIIBjJANBgkqhkiG9w0BAQEFAAOCQA8AMIIBCgKCAQEAz9lXz9
MowulOx0W5v1k7GKlsNy7mAgmkz/wZDImBDXez.QZCb8lr0R8iTD3tI0NH2xpkY3b
uqFjdtQTzCmANLyNwTRlsC5GjN/EMlJSCrO/zZM.ig835RXJTP878N/jNW7EzSxb
/zK5OzKJoRbZ4HgZm4NDapMfMcB4jqBdPxoPAqeR.+KTKv1+9m1vvsdKIs5Hm4Sp
O2WolHPw5BCXdu5zleb6ACih7Zpd2cpHFz6ZHC0G1.Of+f//0BzkfSsqWsmUomyJy
WCfLCuX9grslCNlLxw0gcMprdTxlXjcl8z03ZmBCq0.qq5/mUK/tv9R2k8+WuP3a
kzTUIkeHtcp6FVF13D+TwIDAQABoAAAwDQYJKoZIhvcNAQEFBQADggEBAJH7Etuy
B/oQgQeals08mD2U31FfQ/uYqjNxxZpZJSzVLGMAsv9a.pNzaWdfqPdIs+ZZ+gAQ
QkVcXjdbqY3pAf/EeWk+KnuAUjOIPKu3ZBPVbWbXu/Ie7.F1ekQ7TLkFNkHSxHRu
2/bPIByBLRVfWNVXd3Wpq+QxqMqgIjBGaTJM5kuHndYFGj.Xdf4rlGRPyOOWG/Xf
BrKBB3tzpbjCy+cwOIAJFPOT0+86RuJf9Wh+yoM182vlg80.FyEaaa/PMpl3aEcT
BLrZmPx4e7fLWGuIhbgE7/6K0nF99xdGd7JYPHasbcWszxD0Z.oPYm+44g0Onhlj
OWpRiKXcnegrSSuRILaw=-----END CERTIFICATE REQUEST-----.
== Info: upload completely sent off: 952 out of 952 bytes
== Info: HTTP 1.1 or later with persistent connection, pipelining
supported
```

The EST server accepts this request but since a client certificate was not provided for authentication/authorization the EST server responds with the WWW-authenticate header:

```
<= Recv header, 27 bytes (0x1b)
HTTP/1.1 401 Unauthorized
<= Recv header, 75 bytes (0x4b)
WWW-Authenticate: Digest qop="auth", realm="estrealm", nonce="13
41427174"
```

The EST client repeats the request, this time including the requested Authorization header:

```

== Info: SSL connection using AES256-SHA
== Info: Server certificate:
== Info:  subject: CN=127.0.0.1
== Info:  start date: 2012-07-04 18:39:27 GMT
== Info:  expire date: 2013-07-04 18:39:27 GMT
== Info:  common name: 127.0.0.1 (matched)
== Info:  issuer: CN=estExampleCA
== Info:  SSL certificate verify ok.
== Info: Server auth using Digest with user 'estuser'
=> Send header, 416 bytes (0x1a0)
POST /simpleEnroll HTTP/1.1
Authorization: Digest username="estuser", realm="estrealm", nonc
e="1341427174", uri="/simpleEnroll", cnonce="ODc0OTk2", nc=00000
001, qop="auth", response="48a2b671ccb6596adfef039e134b7d5d"
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 OpenS
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/pkcs10
Content-Length: 952

=> Send data, 952 bytes (0x3b8)
-----BEGIN CERTIFICATE REQUEST-----.MIICHjCCAW4CAQAwQTElMCMGA1UE
AxMccmVxIGJ5IGNsaWVudCBpbjBkZWlwIHN0.ZXAgMjEYMBYGA1UEBRMPUE1EOld
pZGdlldCBTTjoyMIIIBIjANBgkqhkiG9w0BAQEF.AAOCaQ8AMIIBCgKCAQEAz9lXz9
MowulOx0W5v1k7GKlsNy7mAgmkz/wZDIImBDXez.QZCb8lr08iTD3tI0NH2xpkY3b
uqFjdtQTzCmANLyNWtR1sC5GjN/EM1JSCrO/zZM.ig835RXJTP878N/jNW7EzSxb
/zK5OzKJoRbZ4HgZm4NDapMfMcB4jqBdPxopAqER.+Ktkv1+9m1vvsdKIs5Hm4Sp
O2WolHPw5BCXdu5zleb6ACih7Zpd2cpHFz6ZHC0G1.Of+F//0BzkFSqWsmUomyJy
WCfLCuX9grs1CNlLxw0gcMprdTxLxjc18z03ZmBCq0.qq5/mUK/tv9R2k8+WuP3a
kzTUIkeHtcp6FVF13D+TwIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBAJH7Etuy
B/oQgQeals08mD2U31FfQ/uYqjNxxZpZJSzVLGMASv9a.pNzaWdfqPdIs+ZZ+gAQ
QkVcXjdbqY3pAf/EeWk+KnuAUjOIPKu3ZBPVbWbXu/Ie7.F1ekQ7TLkFNkHSxHRu
2/bPIByBLRVfWNVXd3wPq+QxqMqgIjBGaTJM5kuHndYFGj.Xdf4rlGRPyOOwG/Xf
QrKBB3tzpbJCy+cwOUAJFPOTO+86RUjF9Wh+yoM182vlg80.FyEaaA/PMpl3aEcT
BlRZmPx4e7FLwGIhbgE7/6K0nF99xdGd7JYPHasbcWszxD0Z.oPYm+44g0gOnhlj
OWpRiKXcnngSSutRILaw=.-----END CERTIFICATE REQUEST-----.

```

The ESTserver uses the username/password to perform authentication/ authorization and responds with the issued certificate:

```

<= Recv header, 38 bytes (0x26)
0000: Content-Type: application/pkcs7-mime
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

<= Recv data, 1200 bytes (0x4b0)
-----BEGIN PKCS7-----.MIIDUQYJKoZIhvcNAQcCoIIDQjCCAz4CAQExADALBg
kqhkiG9w0BBwGggMkMIID.IDCCAgigAwIBAgIBAJANBgkqhkiG9w0BAQUFADAXM
RUWewYDVQQDEwxc3RFeGft.cGxlQ0EwHhcNMTIwNzA0MTgzOTM0WhcNMTMwNzA0
MTgzOTM0WjBBMSUwIwYDVQQD.ExxyZXEGYnkgy2xpZW50IGluIGRlbW8gc3RlcCA
yMRgwFgYDVQQFEw9QSUQ6V2lk.Z2V0IFNOOjIwggEiMA0GCSqGSIb3DQEBAQUAA4
IBDwAwggEKAoIBAQDP2VfP0yjc.6U7HRbm/WTsYqWw3LuYCCaTP/BkMiYEND7NBk
JvyWs7yJMPE0jQ0fbGmRjdu6oWN.21BPMKYA0vI1a1HWwLkaM38QzU1IKs7/NkyK
DzflFclM/zvw3+M1bsTNLFv/Mrk7.MomhFtngEbmbg0Nqkx8xwHiOoF0/Gg8Cp5H
4pOS/X72bW++x0oizkebhKk7ZaiUc./DkEJd27nOV5voAKKHtm13ZykcxPpkcLQb
U5/4X//QHOQVKpayZSibInJYJ8sK5f.2CuzUI2UvHDSBwmt1PEvGNzXzPTdmYEK
rSqrn+ZQr+2/1HaTz5a4/dqTNNQiR4e.1ynoVUWXcP5PAgMBAAGjTTBLMAkGA1Ud
EwQCMAAwHQYDVR0OBBYEFChDQpKEfG9c.e4JaMf8438tb2XOIMB8GA1UdIwQYMBa
AFIR/SsVuU7I5IC+5INpMScsubQ/zMA0G.CSqGSib3DQEBBQUAA4IBAQAn42mIVG
piaY4yqFD0F8KyUhKsdNnyKeeISQxP//lp.quIieJzdWSc7bhWZNldSznswCod8B
4eJToQeJLSNb8JBDC849z0tcuyHgN6N/p8z.IwI+hAlfXS9q02OECyFes4Jmzc7r
erE5jtOdGsEDBiscw/A+Kv86wv6BKbagMslQ.51AJyPsL6iBhm7LPFrErJgH2kWN
jDKFH9CcVFjXvgriMrLPFeqQWOpj/2XF+4m+c.f9QP5tSjieHJR1hnYk2tlodfE7
iV4pJ07Mmf3yBf753VSUVybqWiMCd0Lm7oghSX.E2GAxrsU1N+N1odn+gJ2wmXTu
AC2aHt9VPRViov4RRtvoQAxA==.-----END PKCS7-----.

```

#### A.4. Re-Enrollment

The following is an example of a valid /simpleReEnroll exchange. During this exchange the EST client authenticates itself using an existing certificate issued by the EST CA.

Initially this exchange is identical to enrollment using an externally issued certificate for client authentication since the server is not yet aware of the client's intention. As in that example the EST server the server generated "certificate request" includes both the distinguished name of the CA the EST server provides services for ("estExampleCA") and it includes the distinguished name of a trusted third party CA ("estEXTERNALCA").

```

0d 00 00 3d 03 01 02 40 00 37 00 1a 30 18 31 16 ...=...@.7..0.1.
30 14 06 03 55 04 03 13 0d 65 73 74 45 58 54 45 0...U....estEXTE
52 4e 41 4c 43 41 00 19 30 17 31 15 30 13 06 03 RNALCA..0.1.0...
55 04 03 13 0c 65 73 74 45 78 61 6d 70 6c 65 43 U....estExampleC
41                                         A

```

In text format this is:

```

Acceptable client certificate CA names
/CN=estEXTERNALCA
/CN=estExampleCA

```

The EST client provides a certificate issued by "estExampleCA" in the certificate response and the TLS handshake proceeds to completion. The EST server accepts the EST client certificate for authentication and accepts the EST client's POSTed certificate request.

The rest of the protocol traffic is effectively identical to a normal enrollment.

#### A.5. Server Key Generation

The following is an example of a valid /serverKeyGen exchange. During this exchange the EST client authenticates itself using an existing certificate issued by the CA the EST server provides services for.

The initial TLS handshake is identical to the enrollment example handshake. The HTTP POSTed message is:

```

POST /serverKeyGen HTTP/1.1
User-Agent: curl/7.24.0 (i686-pc-linux-gnu) libcurl/7.24.0 Opens
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/pkcs10
Content-Length: 968

```

```

=> Send data, 968 bytes (0x3c8)
-----BEGIN CERTIFICATE REQUEST-----.MIICkzCCAXsCAQAwTjEyMDAGA1UE
AxMpc2VydMvYs2V5R2VuIHJlcSBieSBjbGll.bnQgaW4gZGVtbyBzdGVwIDUxGDA
WBgNVBAUTD1BJRDpXaWRnZXQgU046NTCCASIw.DQYJKoZIhvcNAQEBBQADggEPAD
CCAQoCggEBAMnlUlq0ag/fDAVhLgrXead6WtZw.Y2rVGev5saWirer2n0OzghB59
uJBxyp0DYBYqZRuoRF0FTL1ZZTMAzxivge0ecA.ZcoR46jwSBocemT1jkwFyAER
t9Q2EwdnJLIPo/Ib2PLJNb4Jo8NNKmtg55BgIVi.vkIB+rMtLeYRUVL0RUaBAqX
FmtXRDceVFIEY24iUQw6vESGJKpArht592aT8lyap.24bZovuG19dd5xtTX3j37K
x49SlkUvLSpD6ZavIFAZn7Yv19LBKHvRIemybUo294.QeLb/VYP10+EathV/igiX
1DHqLUZCZp5SdyUXUwZPatFboNwEVR0R3MJwVECAwEA.AaAAMA0GCSqGSIb3DQEB
BQUAA4IBAQAqhHezK5/tvbXleHO/aTBVY091414NM+WA.wJcnS2UaJYScPBqlYK/
gij+dgAtFE+5ukAj56t7HnooI4EFo9r8jqChewx7iLZYh.JDxo4hW0sAvHV+Iziy
jkhJNDHBIqGM7Gd5f/2VJLEPQPmwnOL5P+204eQC/QeEYc.bAmfHOS8b/ZH09/9T
PeaeQpjspjOui/100OuLE8KvU3FM0sXMYt1Va0A0jxz1+5k.EiEJo+ltXsQwdP0H
csoTNBN+j3K18omJQS0e91X8v0xkMWYhUtonXD0YZ6SO/B9c.AE6GTADHA/xpSvA
cqlWa+FHxjwEMXdmViHvMUywo31fDZ/TUvCPX.-----END CERTIFICATE REQUE
ST-----.

```

Because the DecryptKeyIdentifier attribute is not included in the request the response does not include additional encryption beyond the TLS session. The EST server response is:

```

<= Recv header, 17 bytes (0x11)
HTTP/1.1 200 OK
<= Recv header, 16 bytes (0x10)
Status: 200 OK
<= Recv header, 67 bytes (0x43)
Content-Type: multipart/mixed ; boundary=estServerExampleBoundar
y
== Info: no chunk, no close, no size. Assume close to signal end
<= Recv header, 2 bytes (0x2)

```

```

<= Recv data, 3234 bytes (0xca2)
This is the preamble. It is to be ignored, though it is a handy
place for estServer to include an explanatory note including con
tact or support information.--estServerExampleBoundary.Content-
Type=application/pkcs8.-----BEGIN PRIVATE KEY-----.MIIEvQIBADAN
BgkqhkiG9w0BAQEFAASCbKcwgSjAgEAAoIBAQC0781l7tri0yii.Mb9ZZYch8ze
izXrjMPF/Rxoz2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSM.tzxn41+9tI
sVDkAe4FyzN0hLd/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImDV3blv.aMUKDSJhV
bQ+z/GlWlTRx3iWi5CMHYb+lpJXPTJz/GuWr/b/+Efqwz2ZlwGcj4Dx.Igbx9vG0

```

mftIIXM4TUX28KBbaLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhg1FU8.DQiQEki  
nn66GPMtm1SNgitxFxWouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhk.g0gMIQ  
TXAgMBAECggEANlrz8XNX/lxBELixK0H83o4aYKYqDKZfZkUN8hU33xpu.Y/0sc  
VbLbu46WzysoIfJFYUC+zFJnbMCCOPjGbI/4NWkEqc9TAlKz+wDo+hf5bf0.ypFr  
EmikHk8R3fKpnvKi69ldw0iYnqcFVhq7VtGrSmJcy6Hckwbk7EBoUZGL0wtp.xl0  
6XlHksAvn8+75qoWzsNhi7S/L0IVCVLbUaV3hodTHlH5M4daFbqyRWD7UiPKt.Q3  
hdwlrpyVZg8ZbBFp0Ej4f9GdRaq88SIKMKCDu3t9ibn/v1kEte+PxhuwyW+d0o.h  
kKSEW0yLKczQm5tjSPq0UVzPBkLJACUnFAi+a4AQKBgQDu6VLH2eYoTjPPTyAv.  
vOJnNWP7oMzyJ4/eFqdE9m+2Ajm/0qaMY95ftZ+GpEKggvC6Z5DFevEmgH4Sg2+G  
.gFd93diyRPScVbNE8SmpXxLPU2UoykVmICuQZzLDNE18B3buxAm2GJ219NenZOe  
c.jPMOV/IcG1aLzTqQssL3zo/0gQKBgQDB40lpg3EBggTJ/+dlkLHUw8c7Pe3UyL  
kS.VxVsyQwioYt8xMeCWuPvPNFc0jCW53KN/YSpCVjpttKGsPtLibMlKYKgasEqg  
cvl.Vb50FtA/jNAP3mdAgCzBn6IF1NhVQe2dclo5puZ0gO38HDWq7EtqSi9Q0JSM  
g3YC.QNcOORptVwKBgQCHrCafaYWDhA11/+g2U9x6Yd56iff43rCbnV+2EQCvaqQ  
i49xC.w4AH+Bs0mdlgt5unL6MOEmgZxkRR/SP7TKzixHYHnpMOqLhaQV24Wk5TQH  
ek92D7.wu8aXRB9vBj4g0CuDNO6/jWpm/KenXXN+Fka3ySVg4zdbVmBzJJdqYckg  
QKBgFXS.zSBzGgwz1/F7AaDZK49mlwPnhyeBb0OqHwbX/LI71rZ1mWef+nSF9Juh  
/Y77B5/J.UPd09vgGgS00nRk0LIRP2s5OU5IQgQTVLv8a1UmbVgI+KX511Yi5yM  
ztEwRcjEX.VM9ejXeXN0I57pvqG/xCOK3K12eYLh4TO9/E8WjjAoGAA1mqUV4Hnf  
4yvF1rydMp.fpvWekiIRE33iEbYZNATYhsl7uxwn760pqVifkq2DSrZeYm4+lw9  
jwWmtUoPzpg.CJYMoG1846nhiZrbbJ5b5twoLV6GRmkk/CfOxPXNzCtSoQA86HHq  
7rRdhXSau/bY.EXc91tnhLjFzZxdBgrd+f4k=-----END PRIVATE KEY-----.  
--estServerExampleBoundary.Content-Type: application/pkcs7-mime.  
.-----BEGIN PKCS7-----.MIIDPAYJKoZIhvcNAQcCoIIDLTCCAYkCAQExADALB  
gkqhkiG9w0BBwGgggMPMIID.CzCCAFogAwIBAgIBBTANBgkqhkiG9w0BAQUFADAX  
MRUwEwYDVQQDEwxc3RFeGFt.cGxlQ0EwHhcNMTIwNzA0MTgzOTM2WhcNMTMwNzA  
0MTgzOTM2WjAsMSowKAYDVQQLD.EyFzZXJ2ZXJzaWRlIGtleSBnZW51cmF0ZWQgc  
VzcG9uc2UwgGEiMA0GCSqGSIb3.DQEBAQUAA4IBDwAwggEKAoIBAQC078117tri0  
yiiMb9ZZYch8zeizXrjMPF/Rxoz.2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+  
DsSMtzzn4l+9tIsVDkAe4FyzN0hL.d/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImD  
V3blvaMUKDSJhVbQ+z/G1W1TRx3iW.i5CMHYb+lpJXPTJz/GuWr/b/+Efqwz2Zlw  
Gcj4DxIgbx9vG0mftIIXM4TUX28KBb.aLgJbalsiuOx3C2bEyaSPerdzqgvXFHGG  
Ahg1FU8DQiQEkin66GPMtm1SNgitxF.xWouFqpsax5MWn/i52TfEaF2PNThOuzK  
tilweJhkg0gMIQTXAgMBAAGjTTBLMAkG.A1UdEwQCMAAwHQYDVRO0BBYEFlylcnQ  
0D5xTfRdayv+0GDULR2+EMB8GA1UdIwQY.MBAFIR/SsVuU7I5IC+5INpMScsubQ  
/zMA0GCSqGSIb3DQEBAQUAA4IBAQButIeM.DB9PkwlGGe7zqvUWVD8y99zowwV6A  
rAOXWX+JO0bihgMtZaUfvPCX/LhZVEKDAki.W5orjAEvIu10b6l38ZzX2oyJgkYy  
Mmbb141zTsRyjiqFw9j1PXxwgZvhwcaCF4b7.eDUUBQIEZg3AnkQrEwnHR5oVIN5  
8qo0P7PSKC3V13H6DlQh3y7w87nN12923/wk0.v/bS3lv7lDX3HdmbQD1r2KPtBs  
JGF4jMdstT7FTx32ZFKObycbK7WJ4LHYtNJDci.4ixf+B0S3D6Zbf1cXj80/W+jC  
GvU0+4SV3cgEXFE5VQvXd8x40W4h0dTSkQCDPOS.nPj4Dl/PsLqX3lDboQAxAA=  
.-----END PKCS7-------estServerExampleBoundary--.This is the ep  
ilogue. It is also to be ignored..

In text format this is:

HTTP/1.1 200 OK  
Status: 200 OK

Content-Type: multipart/mixed ; boundary=estServerExampleBoundary

This is the preamble. It is to be ignored, though it is a handy place for estServer to include an explanatory note including contact or support information.

--estServerExampleBoundary

Content-Type=application/pkcs8

-----BEGIN PRIVATE KEY-----

MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCwggSjAgEAAoIBAQC078117tri0yii  
Mb9ZZYch8zeizXrjMPF/Rxoz2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSM  
tzxn4l+9tIsVDkAe4FyzN0hLd/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImDV3blv  
aMUKDSJhVbQ+z/G1W1TRx3iWi5CMHYb+lpJXPTJz/GuWr/b/+Efqwz2ZlwGcj4Dx  
Igbx9vG0mftIIXM4TUX28KBbaLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhglFU8  
DQiQEkin66GPMtm1SNgitxFxWouFqpsax5MWn/i52TfEaF2PNTThOuzKtilweJhk  
g0gMIQTXAGMBAAECggEANlrz8XNX/lxBELixK0H83o4aYKYqDKZfZkUN8hU33xpu  
Y/0scVbLbu46WzysoIfJFYUC+zFJnbMCCOPjGbI/4NWKEqc9TAlKz+wDo+hf5bf0  
ypFrEmikHk8R3fKpNvKi69ldw0iYnqcFVhQ7VtGrSmJcy6Hckwbk7EBoUZGL0wtp  
xl06XlhksAvn8+75qoWzsNhi7S/L0IVCVLbUaV3hodTH1H5M4daFbqyRWD7UiPKt  
Q3hdwlrpyVZg8ZbBFp0Ej4f9GdRaq88SIKMKCDu3t9ibn/vlkEte+PxhuwyW+d0o  
hkKSEW0yLKCzQm5tUjsPq0UVzPBkLJACUnFAi+a4AQKBgQDu6VLH2eYoTjPPTyAv  
vOJnNWP7oMzyJ4/eFqde9m+2Ajm/0qaMY95ftZ+GpEKggvC6Z5DFevEmgH4Sg2+G  
gFd93diyRPScVbNE8SmpXxLPU2UoykVmICuQZzLDNE18B3buxAm2GJ219NenZOec  
jPMOV/IcG1aLzTqQssL3zo/0gQKBgQDB40lpg3EBggTJ/+dlkLHUW8c7Pe3UyLkS  
VxVsyQwioYt8xMeCWuPvPNFcoJcW53KN/YSpCVjpttKGsPtLibM1KYKgasEqgcVl  
Vb50ftA/jNAP3mdAgCzBn6IF1NhVQe2dclo5puZ0g038HDWq7EtqSi9Q0JSMg3YC  
QNcOORptVwKBgQChrcafaYWDhA11/+g2U9x6Yd56iff43rCbnV+2EQCvaqQi49xC  
w4AH+BsoMdlgt5unL6MOEmgZxkRR/SP7TKzixHYHnpMOqLhaQV24Wk5TQHEk92D7  
wu8aXRB9vBj4g0CuDNO6/jWpm/KenXXN+Fka3ySVg4zdbVmBzJJdqYckgQKBgFXS  
zSBzGgwz1/F7AaDZK49m1wPnhyeBb0OqHwbX/LI71rZ1mWef+nSF9Juh/Y77B5/J  
UPdO9vgGgS00nRk0LIRP2s5OU5IQgQTVLvf8a1UmbVgI+KX511Yi5yMztEwRcjEX  
VM9ejXeXN0I57pvqG/xCOK3K12eYlH4TO9/E8WjjAoGAA1mqUV4Hnf4yvF1rydMp  
fpvoWekiiRE33iEbYZNATYhsl7uxwn760pqVifkq2DSrZeYm4+lw9jwWMTUoPzpg  
CJYMoG1846nhiZrbbJ5b5twoLV6GRmkk/CfOxPXNzCtSoQA86HHq7rRdhXSau/bY  
EXc91tnhLjFzZxdBgrd+f4k=

-----END PRIVATE KEY-----

--estServerExampleBoundary

Content-Type: application/pkcs7-mime

-----BEGIN PKCS7-----

MIIDPAYJKoZIhvcNAQcCoIIDLTCCAYkCAQEExADALBgkqhkiG9w0BBWGGggMPMIID  
CzCCAfOgAwIBAgIBBTANBgkqhkiG9w0BAQUFADAXMRUwEwYDVQQDEwxc3RFeGFt  
cGxlQ0EwHhcNMTIwNzA0MTgzOTM2WhcNMTMwNzA0MTgzOTM2WjAsMSowKAYDVQQD  
EyFzZXJ2ZXJzaWRlIGt1eSBnZW51cmF0ZWQgcMvzcG9uc2UwgGEmA0GCSqGSIb3  
DQEBQUAA4IBDwAwggEKAoIBAQC078117tri0yiiMb9ZZYch8zeizXrjMPF/Rxoz  
2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSMtzxn4l+9tIsVDkAe4FyzN0hL  
d/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImDV3blvaMUKDSJhVbQ+z/G1W1TRx3iW  
i5CMHYb+lpJXPTJz/GuWr/b/+Efqwz2ZlwGcj4DxIgbx9vG0mftIIXM4TUX28KBb

```

aLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhglFU8DQiQEkin66GPMtm1SNgitxF
xWouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhkg0gMIQTXAgMBAAGjTTBLMAkG
A1UdEwQCMAAwHQYDVR0OBByEFlylcQN0D5xTfRdayv+0GDULR2+EMB8GA1UdIwQY
MBaAFIR/SsVuU7I5IC+5INpMScsubQ/zMA0GCSqGSIB3DQEBBQUAA4IBAQBUTieM
DB9PkwlgGe7zqvUWVD8y99zowwV6ArAOXWX+JO0bihgMtZaUfvPCX/LhZVEKDAki
W5orjAEvIu10b6138ZzX2oyJgkYyMmbb14lzTsRyjiqFw9j1PXxwgZvhwcaCF4b7
eDUUBQIeZg3AnkQrEwnHR5oVIN58qo0P7PSKC3V13H6D1Qh3y7w87nN12923/wk0
v/bS3lv7lDX3HdmbQDlr2KPtBsJGF4jMdstT7FTx32ZFKObycbK7WJ4LHytNJDci
4ixf+B0S3D6ZbflcXj80/W+jCGvU0+4SV3cgEXFE5VQvXd8x40W4h0dTSkQCDPOS
nPj4Dl/PsLqX3lDboQAxA==

```

```
-----END PKCS7-----
```

```
--estServerExampleBoundary--
```

This is the epilogue. It is also to be ignored.

#### A.6. CSR Attributes

The following is an example of a valid /CSRAttrs exchange. During this exchange the EST client authenticates itself using an existing certificate issued by the CA the EST server provides services for.

The initial TLS handshake is identical to the enrollment example handshake. The HTTP GET request:

```
GET /CSRAttrs HTTP/1.1
```

```
User-Agent: curl/7.22.0 (i686-pc-linux-gnu) libcurl/7.22.0 OpenS
SL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
```

```
Host: 127.0.0.1:8085
```

```
Accept: */*
```

In response the server provides suggested attributes that are appropriate for the authenticated client:

```
<= Recv header, 36 bytes (0x24)
```

```
Content-Type: application/csrattrs
```

```
== Info: no chunk, no close, no size. Assume close to signal end
```

```
<= Recv header, 2 bytes (0x2)
```

```
<= Recv data, 33 bytes (0x21)
```

```
0000: MBQGBysGAQEBAARYGCSqGSIB3DQEJBw==.
```

Authors' Addresses

Max Pritikin (editor)  
Cisco Systems, Inc.  
510 McCarthy Drive  
Milpitas, CA 95035  
USA

Email: pritikin@cisco.com

Peter E. Yee (editor)  
AKAYLA, Inc.  
7150 Moorland Drive  
Clarksville, MD 21029  
USA

Email: peter@akayla.com

Dan Harkins (editor)  
Aruba Networks  
1322 Crossman Avenue  
Sunnyvale, CA 94089-1113  
USA

Email: dharkins@arubanetworks.com



INTERNET-DRAFT  
Intended Status: Informational  
Expires: September 12, 2013

Stefan Santesson  
(3xA Security)  
March 11, 2013

Authentication Context Certificate Extension  
draft-santesson-auth-context-extension-04

Abstract

This document defines an extension to certificates according to [RFC5280]. The extension defined in this document holds data about how the certificate subject was authenticated by the Certification Authority who issued the certificate where this extension appears.

This document also defines one data structure for inclusion in this extension that designed to hold information when the subject is authenticated using a SAML assertion [SAML].

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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/lid-abstracts.html>

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

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents  
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|       |  |    |
|-------|--|----|
| 1     | Introduction . . . . .   | 3  |
| 1.1   | Terminology . . . . .  | 4  |
| 1.2   | Deployment . . . . .   | 4  |
| 2.    | Authentication Context Extension Syntax . . . . .                    | 5  |
| 3     | SAML Authentication Context Information . . . . .                    | 6  |
| 3.1   | contextInfo Data Structure . . . . .                                 | 6  |
| 3.1.1 | AuthContextInfo Element . . . . .                                    | 6  |
| 3.1.2 | IdAttributes Element . . . . .                                       | 8  |
| 4     | Security Considerations . . . . .                                    | 10 |
| 5     | IANA Considerations . . . . .  | 10 |
| 6     | References . . . . .   | 10 |
| 6.1   | Normative References . . . . .                                       | 10 |
| 6.2   | Informative References . . . . .                                     | 11 |
|       | Appendix A - ASN.1 modules . . . . .                                 | 11 |
| A.1   | ASN.1 1988 Syntax . . . . .  | 11 |
| A.2   | ASN.1 2008 Syntax . . . . .  | 12 |
|       | Appendix B - SAML Authentication Context Info XML Schema . . . . .   | 13 |
| B.1   | XML Schema . . . . .   | 13 |
|       | Appendix C - SAML Authentication Context Info XML Examples . . . . . | 15 |
| C.1   | Complete context information and mappings . . . . .                  | 15 |
| C.2   | Only mapping information without SAML attribute values . . . . .     | 16 |
| C.3   | Authentication context and serialNmber mapping . . . . .             | 17 |
|       | Authors' Addresses . . . . .   | 18 |

## 1 Introduction

The primary purpose of this document is to provide a mechanism that allows an application to understand information that express the identity of a subject in a certificate that is stored either in a subject field attribute, as a subject alternative name or in a subject directory attribute.

This addresses some needs that may arise when issuing a certificate from an existing non-certificate based identity infrastructure where the certificate subject already has an authenticated identity composed of a set of attributes, or so called claims, that differ from the attributes that are commonly used to express the identity of a certificate subject.

A typical scenario for this is when the source of user authentication and user identity is based on a SAML federation, where the subject presents a SAML assertion in exchange of a certificate that can be traced back to that subjects SAML assertion, both with regard to identity attributes and with regard to level of assurance with which the subject was authenticated by its Identity Provider.

A reason to issue such certificate may arise if the subject needs a certificate to sign a document, where the Certification Authority is authenticating the user by means of a SAML assertion when issuing that signature certificate.

If that signature certificate need to conform to certificate profiles, such as [RFC3739], then this certificate may have to use a separate set of attributes to express the subject identity, as well as different formats for the attribute value, than the set of attributes obtained from the SAML assertion.

The extension defined in the document makes it possible to extract information about the authentication context applied when authenticating the subject for the purpose of issuing a certificate. This may include information such as:

- o The Identity Provider which authenticated the subject.
- o The level of assurance with which the subject was authenticated.
- o The trust framework where this level of assurance was defined.
- o A unique reference to the authentication instant
- o A mapping table between the subject attributes obtained from the SAML assertion used to authenticate the subject, and the subject identity information placed in the issued certificate.

One scenario where this information may be useful is when a user logs in to a service using SAML credentials, where the same user at some

stage is required to sign some information. The service may need to verify that the signature was created by the same user that logged on to the service. This is only possible today using out-of-band knowledge about the CA that issued the certificate and it's practices. This is is however hard to scale and maintain using a large number of service providers, identity providers and CAs.

The defined extension provides better scalability since it only requires the service provider to maintain a list of trusted CA:s. All other information about the relationship between the certificate subject, and the SAML authenticated subject is available in the certificate.

### 1.1 Terminology

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 [RFC2119].

### 1.2 Deployment

#### EDITORS NOTE:

[This section provided information for better understanding the rationale of the extension. This section can be deleted is the document is published]

The extension defined in this draft has been defined and deployed in the National Swedish Identity infrastructure Eid 2.0 which is based on SAML federated identity. The Swedish infrastructure will go live during 2013 and will provide secure identification of citizens in Swedish government services. A central requirement in these government services is to allow citizens to sign various documents, representing a wide range of declarations and applications.

A central part of this infrastructure is therefore to use centralized signature services that allows citizens to sign using their SAML credentials. As service providers authenticate and understands user identities only under a SAML context within this national infrastructure, this extension allows Service Providers to determine whether a presented signature matches a particular user and whether it meets the security requirements of the service.

Through information provided in this extension a service provider may for example get notice that the user logged on using one level of assurance, but presented a signature which certificate was issued using a certificate obtained using a lower level of assurance procedure, and thus reject the signature.

This extension is therefore fundamental to the function of the Swedish Eid 2.0 infrastructure.

## 2. Authentication Context Extension Syntax

The Authentication Context extension has the following syntax:

```
AuthenticationContexts ::= SEQUENCE SIZE (1..MAX) OF
                           AuthenticationContext

AuthenticationContext ::= SEQUENCE {
    contextType      UTF8String,
    contextInfo      UTF8String OPTIONAL
}
```

This extension holds a sequence of AuthenticationContext information. When present, this extension MUST include at least one AuthenticationContext.

The type of authentication context defined in AuthenticationContext is identified by the contextType which MUST contain a URI that identifies the context type as well as an XML Schema name space [Schema1] and [Schema2] for associated context information. The optional authentication context information is carried as an XML [XML] string in contextInfo in accordance with the identified XML Schema.

The XML data format is used mainly to allow context information to be extracted and processed in applications that lacks ASN.1 processing capabilities. XML is easy to deserialize into various data objects both in application and web environments for further comparison with the characteristics of for example SAML authenticated sessions.

This extension MAY be marked critical.

Applications which find an authentication context information type they do not understand MUST ignore it if the extension is non critical, and MUST reject the certificate if the extension is critical. If an application requires that an authentication context exist, and either the extension is absent or none of the provided authentication contexts can be used MUST fail validation of the end user certificate.

This document defines one authentication context information type (Section 3) that is used to provide information about SAML based authentication of the subject as part of the certificate issuance

process. Other documents can define other authentication context information types.

### 3 SAML Authentication Context Information

The SAML Authentication context information provides a contextType type that can be used to carry information about SAML based authentication of the certified subject as part of the certificate issuance process.

The data carried in this authentication context information type is identified by the following XML Schema name space:

`http://id.elegnamnden.se/auth-cont/1.0/saci`

When this URI is specified as contextType, then associated XML data MUST be provided in contextInfo

#### 3.1 contextInfo Data Structure

The data provided in contextInfo SHALL contain UTF-8 encoded XML in accordance with the XML schema provided in Appendix B. The XML document string in contextInfo MUST NOT include an XML header. That is, the XML document string contains only the root element `<SAMLAuthContext>` with it's child elements `<AuthContextInfo>` and `<IdAttributes>`.

The `<AuthContextInfo>` and `<IdAttributes>` elements are outlined in the following subsections.

##### 3.1.1 AuthContextInfo Element

The `<AuthContextInfo>` element MAY be present. This element contains the following attributes:

|                       |  |
|-----------------------|--|
| IdentityProvider      | (required): The SAML EntityID of the Identity Provider which authenticated the subject.  |
| AuthenticationInstant | (required): Date and time when the subject was authenticated, expressed according to section 3.3.  |
| AuthnContextClassRef  | (required): A URI identifying the AuthnContextClassRef that is provided in the AuthnStatement of the Assertion that was used to authenticate the subject. This URI |

|              |  |
|--------------|--|
|              | identifies the context and the level of assurance associated with this instance of authentication. |
| AssertionRef | (optional): A unique reference to the SAML Assertion   |
| ServiceID    | (optional): An arbitrary identifier of the service that verified the SAML assertion.               |

The <AuthContextInfo> element may hold any number of child elements of type any (processContents="lax"), providing additional information according to local conventions. Any such elements MAY be ignored if not understood.

### 3.1.2 IdAttributes Element

The <IdAttributes> element MAY be present. This element holds a sequence of one or more <AttributeMapping> elements, where each <AttributeMapping> element holds mapping information about one certificate subject attribute or name form present in the certificate.

Each <AttributeMapping> element MUST specify the following attributes:

- |       |  |
|-------|--|
| Type  | A string holding one of the enumerated values "rdn", "san" or "sda", specifying the type of certificate attribute or name form for which mapping information is provided:  |
| "rdn" | Mapping information is provided for an attribute in a Relative Distinguished Name located in the subject field.  |
| "san" | Mapping information is provided for a name in the Subject Alternative Name extension of the certificate.   |
| "sda" | Mapping information is provided for an attribute in the Subject Directory Attributes extension.  |
| Ref   | A reference to the specific attribute or name field. This reference is dependent on the value of Type in the following way:  |
| "rdn" | REF holds a string representation of the OID of the relative distinguished name attribute.   |
| "sda" | REF holds a string representation of the OID of the subject directory attribute attribute.   |
| "san" | REF holds a string representation of the explicit tag number of the Subject Alternative Name type (e.g. "1" = e-mail address (rfc822Name) and "2" = dNSName). If the SubjectAlternative name is an otherName, then the Ref holds a string representation of the OID defining the otherName form. |

String representations of object identifiers (OID) in the Ref attribute MUST be represented by a sequence of integers separated by a period. E.g. "2.5.4.32". This string MUST NOT contain any white-space or line breaks.

Each <AttributeMapping> element MUST contain a <saml:Attribute> element as defined in [SAML]. This SAML attribute element MUST have a Name attribute (specifying its type), MAY have other attributes and

MAY have zero or more <saml:AttributeValue> child elements. A present SAML attribute with absent attribute value limits mapping to the type of SAML attribute that was used to obtain the value stored in the referenced certificate subject attribute or name form, without duplicating the actual attribute value.

If an attribute value is present in the SAML attribute, then the value stored in the certificate in the referenced attribute or name form MAY differ in format and encoding from the present SAML attribute value. For example, a SAML attribute value can specify a country expressed as "Sweden" while this country value is stored in the certificate in a countryName attribute using the two letter country code "SE".

Several <AttributeMapping> elements MAY be present for the same certificate subject attribute or name form if the certificate contains multiple instances of this attribute or name form where their values were obtained from different SAML attributes. But in such case it is not defined which present subject attribute or name form that maps to which SAML attribute. A certificate using application MAY attempt to determine this by comparing attribute values stored in this extension with attribute or name values present in the certificate, but this specification does not define any explicit matching rules that would guarantee an unambiguous result.

The <AttributeMapping> element may hold any number of child elements of type any (processContents="lax"), providing additional information according to local conventions. Any such elements MAY be ignored if not understood.

Note: The <AttributeMapping> element is designed to provide mapping between SAML attributes and certificate subject attributes and name forms where there is a distinct and clear relationship between relevant SAML attributes and corresponding certificate attributes and name forms. This does not cover all aspects of complex mapping situations where e.g. more than one SAML attribute maps to the same certificate attribute or when structured multi valued attributes are split into a range of other attributes and name forms. Such complex mapping situations MAY be covered by extending this XML Schema or by defining a more versatile context information schema.

#### 4 Security Considerations

This extension allows a CA to outsource the process to identify and authenticate a subject to another trust infrastructure in a dynamic manner that may differ from certificate to certificate. Since the authentication context is explicitly declared in the certificate, one certificate may be issued with a lower level of assurance than another.

This means that the relying party need to be aware of the certificate policy under which this CA operates in order to understand when the certificate provides a level of assurance with regard to subject authentication that is higher than the lowest provided level. A relying party that is not capable of understanding the information in the authentication context extension MUST assume that the certificate is issued using the lowest allowed level of assurance declared by the policy.

#### 5 IANA Considerations

This document contains no actions for IANA.

#### 6 References

##### 6.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3739] Santesson, S., Nystrom, M., and T. Polk, "Internet X.509 Public Key Infrastructure: Qualified Certificates Profile", RFC 3739, March 2004.
- [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.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.
- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, June 2010.
- [SAML] Scot Cantor, John Kemp, Rob Philpott, Eve Maler, "Assertions and Protocols for the OASIS Security

Assertion Markup Language (SAML) V2.0", OASIS Standard, 15 March 2005.

[XML] Extensible Markup Language (XML) 1.0 (Fifth Edition), <http://www.w3.org/TR/REC-xml/#sec-element-content>, W3C Recommendation 26 November 2008.

[Schema1] H. S. Thompson et al. XML Schema Part 1: Structures, <http://www.w3.org/TR/xmlschema-1/>, W3C Recommendation, May 2001.

[Schema2] P. V. Biron et al. XML Schema Part 2: Datatypes. <http://www.w3.org/TR/xmlschema-2/>, W3C Recommendation, May 2001.

## 6.2 Informative References

No informational references

## Appendix A - ASN.1 modules

This appendix includes the ASN.1 modules for the Authentication Context extension. Appendix B.1 includes an ASN.1 module that conforms to the 1998 version of ASN.1. Appendix B.2 includes an ASN.1 module, corresponding to the module present in B.1, that conforms to the 2008 version of ASN.1. Although a 2008 ASN.1 module is provided, the module in Appendix B.1 remains the normative module as per policy adopted by the PKIX working group for certificate related specifications.

### A.1 ASN.1 1988 Syntax

ACE-88

```
{iso(1) member-body(2) se(752) e-legnamnden(201)
 id-mod(0) id-mod-auth-context-88(1)}
```

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS

-- Certificate Extensions

Extensions

FROM PKIX1Explicit88 { iso(1) identified-organization(3)

```
dod(6) internet(1) security(5) mechanisms(5) pkix(7)
id-mod(0) id-pkix1-explicit(18) };
```

```
-- Authentication Context Extension
```

```
AuthenticationContexts ::= SEQUENCE SIZE (1..MAX) OF
    AuthenticationContext
```

```
AuthenticationContext ::= SEQUENCE {
    contextType      UTF8String,
    contextInfo      UTF8String OPTIONAL
}
```

```
e-legnamnden      OBJECT IDENTIFIER ::= { iso(1) member-body(2)
                                         se(752) 201 }
id-eleg-ce        OBJECT IDENTIFIER ::= { e-legnamnden 5 }
id-ce-authContext OBJECT IDENTIFIER ::= { id-eleg-ce 1 }
```

```
END
```

## A.2 ASN.1 2008 Syntax

```
ACE-08
```

```
{iso(1) member-body(2) se(752) e-legnamnden(201)
 id-mod(0) id-mod-auth-context-08(2)}
```

```
DEFINITIONS EXPLICIT TAGS ::=
BEGIN
IMPORTS
```

```
Extensions{}, EXTENSION
FROM PKIX-CommonTypes-2009 -- From [RFC5912]
{iso(1) identified-organization(3) dod(6) internet(1) security(5)
 mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57)}
```

```
ext-AuthenticationContext EXTENSION ::= { SYNTAX
    AuthenticationContexts IDENTIFIED BY
    id-ce-authContext }
```

```
AuthenticationContexts ::= SEQUENCE SIZE (1..MAX) OF
    AuthenticationContext
```

```
AuthenticationContext ::= SEQUENCE {
    contextType      UTF8String,
    contextInfo      UTF8String OPTIONAL
}
```

```
e-legnamnden      OBJECT IDENTIFIER ::= { iso(1) member-body(2)
                                     se(752) 201 }
id-eleg-ce        OBJECT IDENTIFIER ::= { e-legnamnden 5 }
id-ce-authContext OBJECT IDENTIFIER ::= { id-eleg-ce 1 }
```

END

## Appendix B - SAML Authentication Context Info XML Schema

This appendix section B.1 includes an XML Schema ([Schema1] and [Schema2]) for the SAML Authentication context information defined in section 3.

IMPORTANT NOTE: The XML Schema in B.1 specifies a URL on row 9 and 10 to the SAML schemaLocation (<http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-2.0.xsd>), which is too long to fit into one row and therefore contains a line-break. This line-break has to be removed before this schema can be successfully compiled.

### B.1 XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  targetNamespace="http://id.elegnamnden.se/auth-cont/1.0/saci"
  xmlns:saci="http://id.elegnamnden.se/auth-cont/1.0/saci"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">

  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
    schemaLocation="http://docs.oasis-open.org/security/saml/v2.0/
saml-schema-assertion-2.0.xsd"/>

  <xs:element name="SAMLAuthContext"
    type="saci:SAMLAuthContextType"/>
  <xs:complexType name="SAMLAuthContextType">
    <xs:sequence>
      <xs:element ref="saci:AuthContextInfo" minOccurs="0"/>
      <xs:element ref="saci:IdAttributes" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="AuthContextInfo"
    type="saci:AuthContextInfoType"/>
  <xs:complexType name="AuthContextInfoType">
    <xs:sequence>
      <xs:any processContents="lax"
```

```
        minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="IdentityProvider"
              type="xs:string" use="required"/>
<xs:attribute name="AuthenticationInstant"
              type="xs:dateTime" use="required"/>
<xs:attribute name="AuthnContextClassRef"
              type="xs:anyURI" use="required"/>
<xs:attribute name="AssertionRef" type="xs:string"/>
<xs:attribute name="ServiceID" type="xs:string"/>
</xs:complexType>

<xs:element name="IdAttributes" type="saci:IdAttributesType"/>
<xs:complexType name="IdAttributesType">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" minOccurs="1"
                ref="saci:AttributeMapping"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="AttributeMapping"
              type="saci:AttributeMappingType"/>
<xs:complexType name="AttributeMappingType">
  <xs:sequence>
    <xs:element ref="saml:Attribute"/>
    <xs:any processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="rdn"/>
        <xs:enumeration value="san"/>
        <xs:enumeration value="sda"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="Ref" type="xs:string" use="required"/>
</xs:complexType>
</xs:schema>
```

## Appendix C - SAML Authentication Context Info XML Examples

This appendix provides examples of SAML Authentication Context information according to the schema in Appendix B.

## C.1 Complete context information and mappings

This example provides a complete example with authentication context information as well as mapping information for several subject field attributes as well as a subject alt name.

```
<saci:SAMLAuthContext
  xmlns:saci="http://id.elegnamnden.se/auth-cont/1.0/saci"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <saci:AuthContextInfo
    ServiceID="eid2csig"
    AssertionRef="_71b981ab017eb42869ae4b62b2a63add"
    IdentityProvider="https://idp-test.nordu.net/idp/shibboleth"
    AuthenticationInstant="2013-03-05T22:59:57.000+01:00"
    AuthnContextClassRef="http://id.elegnamnden.se/loa/1.0/loa3"/>
  <saci:IdAttributes>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.6">
      <saml:Attribute
        FriendlyName="Country"
        Name="urn:oid:2.5.4.6">
        <saml:AttributeValue xsi:type="xs:string"
          >SE</saml:AttributeValue>
        </saml:Attribute>
      </saci:AttributeMapping>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.5">
      <saml:Attribute
        FriendlyName="Personal ID Number"
        Name="urn:oid:1.2.752.29.4.13">
        <saml:AttributeValue xsi:type="xs:string"
          >200007292386</saml:AttributeValue>
        </saml:Attribute>
      </saci:AttributeMapping>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.42">
      <saml:Attribute
        FriendlyName="Given Name"
        Name="urn:oid:2.5.4.42">
        <saml:AttributeValue xsi:type="xs:string"
          >John</saml:AttributeValue>
        </saml:Attribute>
      </saci:AttributeMapping>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.4">
      <saml:Attribute
```

```

        FriendlyName="Surname"
        Name="urn:oid:2.5.4.4">
        <saml:AttributeValue xsi:type="xs:string"
        >Doe</saml:AttributeValue>
    </saml:Attribute>
</saci:AttributeMapping>
<saci:AttributeMapping Type="rdn" Ref="2.5.4.3">
    <saml:Attribute
        FriendlyName="Display Name"
        Name="urn:oid:2.16.840.1.113730.3.1.241">
        <saml:AttributeValue xsi:type="xs:string"
        >John Doe</saml:AttributeValue>
    </saml:Attribute>
</saci:AttributeMapping>
<saci:AttributeMapping Type="san" Ref="1">
    <saml:Attribute
        FriendlyName="E-mail"
        Name="urn:oid:0.9.2342.19200300.100.1.3">
        <saml:AttributeValue xsi:type="xs:string"
        >john.doe@example.com</saml:AttributeValue>
    </saml:Attribute>
</saci:AttributeMapping>
</saci:IdAttributes>
</saci:SAMLAuthContext>

```

## C.2 Only mapping information without SAML attribute values

This example shows an instance of the SAML Authentication Context information that only provides a mapping table without providing any authentication context information or saml attribute values.

```

<saci:SAMLAuthContext
  xmlns:saci="http://id.elegnamnden.se/auth-cont/1.0/saci"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
  <saci:IdAttributes>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.6">
      <saml:Attribute Name="urn:oid:2.5.4.6"/>
    </saci:AttributeMapping>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.5">
      <saml:Attribute Name="urn:oid:1.2.752.29.4.13"/>
    </saci:AttributeMapping>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.42">
      <saml:Attribute Name="urn:oid:2.5.4.42"/>
    </saci:AttributeMapping>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.4">
      <saml:Attribute Name="urn:oid:2.5.4.4"/>
    </saci:AttributeMapping>
  </saci:IdAttributes>
</saci:SAMLAuthContext>

```

```
<saci:AttributeMapping Type="rdn" Ref="2.5.4.3">
  <saml:Attribute Name="urn:oid:2.16.840.1.113730.3.1.241"/>
</saci:AttributeMapping>
<saci:AttributeMapping Type="san" Ref="1">
  <saml:Attribute Name="urn:oid:0.9.2342.19200300.100.1.3"/>
</saci:AttributeMapping>
</saci:IdAttributes>
</saci:SAMLAuthContext>
```

### C.3 Authentication context and serialNmber mapping

This example shows an instance of the SAML Authentication Context information, which provides authentication context information and mapping information that specifies the source of the data stored in the serialNumber attribute in the subject field.

```
<saci:SAMLAuthContext
  xmlns:saci="http://id.elegnamnden.se/auth-cont/1.0/saci"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <saci:AuthContextInfo
    ServiceID="eid2csig"
    AssertionRef="_71b981ab017eb42869ae4b62b2a63add"
    IdentityProvider="https://idp-test.nordu.net/idp/shibboleth"
    AuthenticationInstant="2013-03-05T22:59:57.000+01:00"
    AuthnContextClassRef="http://id.elegnamnden.se/loa/1.0/loa3"/>
  <saci:IdAttributes>
    <saci:AttributeMapping Type="rdn" Ref="2.5.4.5">
      <saml:Attribute
        FriendlyName="Personal ID Number"
        Name="urn:oid:1.2.752.29.4.13">
        <saml:AttributeValue xsi:type="xs:string">
          200007292386</saml:AttributeValue>
        </saml:Attribute>
      </saci:AttributeMapping>
    </saci:IdAttributes>
  </saci:SAMLAuthContext>
```

Authors' Addresses

Stefan Santesson  
3xA Security AB  
Scheelev. 17  
223 70 Lund  
Sweden  
EMail: sts@aaa-sec.com