PKIX
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2013

M. Pritikin, Ed.
Cisco Systems, Inc.
P. Yee, Ed.
AKAYLA, Inc.
D. Harkins, Ed.
Aruba Networks
July 10, 2012

Enrollment over Secure Transport
draft-ietf-pkix-est-02

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 simple Public Key Infrastructure clients that need to acquire client certificate(s) and associated Certification Authority (CA) certificate(s).

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 January 11, 2013.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a protocol for certificate Enrollment over Secure Transport (EST). EST is designed to be easily implemented by clients and servers using common "off the shelf" PKI, HTTP, and TLS components. An EST server providing certificate management functions is operated by (or on behalf of) a CA or RA. The goal is to provide a small set of functions for certificate enrollment that are simpler to implement and use than full CMP or CMC. While less functional than those protocols, EST satisfies basic needs by providing an easily implemented means for both autonomous devices as well as user-operated computers to request certificates.

The TLS [RFC4346] (or later) protocol is used with a limited set of features of the Certificate Management over CMS (CMC) [RFC5272] to provide the security for EST. CMC "simple" messages are used for certificate requests and responses. EST also allows the optional use of "full" CMC messages if needed, but compliant EST client and server implementations need not support full CMC messages. EST adopts the CMP model for CA certificate rollover, but does not incorporate its syntax or protocol. An EST server supports several means of authenticating a certificate requester, leveraging the layering of the protocols that make up EST. EST servers are extensible in that new requests may be defined which provide additional capabilities not specified in the base RFC. One non-CMC-based extension (requesting of CSR attributes) is defined in this document.

EST works by transporting CMC and other messages securely over an HTTPS transport in which HTTP headers and content types are used in conjunction with TLS security. TCP/IP sits under HTTPS; this document does not specify EST over DTLS or UDP. Figure 1 shows how the layers build upon each other.

EST Layering:

Protocols:

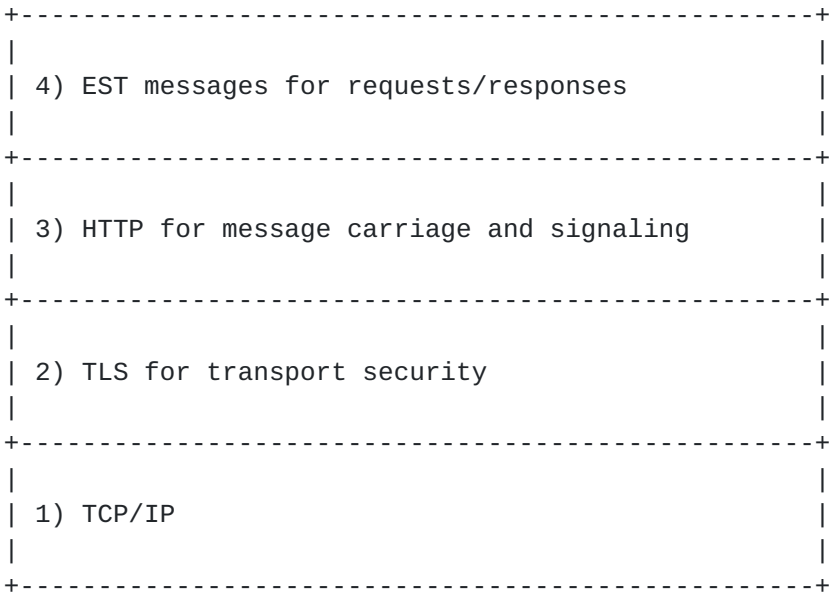

Figure 1

[[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. Requirements Language

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].

2. Operational Scenario Overviews

This EST specification provides a profile of CMC using round-trip communication between the EST client and the EST server in which CMC "simple" messages are transmitted. The basic framework can be extended with additional capabilities that leverage the transport and security features supplied by EST.

The EST server is assumed to be configured with an identity certificate and appropriate policy regarding authenticated clients. An EST server likely communicates with a CA for signing but for simplicity we indicate that a' certificate is signed' as if by the EST server. The EST client is initially configured with only the HTTPS URI of the EST server.

This section illustrates several potential certificate enrollment and rekey scenarios supported by this profile. For clarity the EST client is assumed to perform "Obtaining CA Certificates" before performing other operations.

This section does not intend to place any limits or restrictions on the use of full CMC. Sections 2.1-2.3 very closely mirror the exact text of the Scenarios Appendix of [RFC6403] with such modifications as are appropriate for this profile. (Our thanks are extended to the authors of that document).

2.1. Obtaining CA Certificates

The EST client can request a copy of the current CA certificates.

Following the logic laid out in Section 3.3.1.1 the EST client authenticates and authorizes the EST server. Available options include verifying the EST server URI against the EST server certificate (similar to a common HTTPS exchange), or using a "pinned" copy of the CA certificate. As a fallback the EST client can accept manual authentication performed by the end user (in which case the certificates received are be "pinned" for authenticating future communications with the EST server).

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

2.2. Initial Enrollment

The EST client can enroll with the CA server by submitting an enrollment request to the EST server. Following the logic laid out in Section 3.3.1.1 the EST client authenticates and authorizes the EST server.

Three scenarios for the EST server to authenticate the enrollment requests are:

1. Previously installed signature certificate (e.g., Manufacturer Installed Certificate or 3rd party issued certificate);

2. Username/password distributed out-of-band

3. RA authentication

2.2.1. Previously Installed Signature Certificate

If the EST client has a previously installed signature certificate issued by a trust anchor listed by the EST server during the TLS handshake it can be used to authenticate the request for a new certificate. The EST client responds to the TLS certificate request with the existing certificate as defined for TLS. The EST server will recognize the authorization of the previously installed certificate and issue an appropriate certificate to the EST client.

2.2.2. Username/Password Distributed Out-of-Band

If the EST client did not have a previously installed signature certificate, or if the EST server wishes additional authentication information, the EST server requests the EST client submit a username/password using the HTTP authentication methods.

2.2.3. RA Authentiation

In this scenario the EST client submits the certification request using either the /simpleEnroll or /fullCMC method. The EST server forwards the received request using either CMC or other methods out-of-scope of this document.

2.3. Re-Enrollment

The EST client can renew/rekey an existing client certificate by submitting a re-enrollment request to the EST server. As for initial enrollment the EST server authenticates the client using any combination of the existing client certificate and an HTTP username/password. Because the client specifically requests renew/rekey the EST server can adjust its policy accordingly.

There are two scenarios to support the renew/rekey of clients that are already enrolled. One addresses the renew/rekey of signature certificates and the other addresses the renew/rekey of key establishment certificates. Typically, organizational policy will require certificates to be currently valid to be renewed/rekeyed, and it may require initial enrollment to be repeated when renew/rekey is not possible.

2.3.1. Re-Enrollment of Signature Certificates

When a signature certificate is re-enrolled the existing certificate is used by the EST client for authentication. The EST server uses this information along with any supplimental HTTP authentication information and the certification request itself to determine the parameters of the certificate to issue in response. If there is no

current signature certificate available the EST server can fallback on the HTTP authentication method. The certification request message will include the same Subject/SubjectAltName as the current signature certificate.

2.3.2. Re-Enrollment of Key Establishment Certificates

When a key establishment certificate is re-enrolled an existing signature certificate is used by the EST client for authentication. The EST server uses this information along with any supplimental HTTP authentication information and the certification request itself to determine the parameters of the certificate to issue in response. If there is no current signature certificate available the EST server can fallback on the HTTP authentication method. The certification request message will include the same Subject/SubjectAltName as the current key establishment certificate.

2.4. Server Key Generation

The EST client can request a server generated certificate and keypair. The EST server authenticates the client using any existing client signature certificate and/or HTTP username/password.

2.5. Full CMC messages

Full CMC messages can be transported thus allowing access to functionality not provided by the simple CMC message. "Full" CMC messages are as defined in Sections 3.2 and 4.2 of [RFC5272]. Support for full CMC message transport is optional for EST clients and servers.

2.6. CSR Attributes Request

Prior to sending an enrollment request to an EST server, an EST client may request that the EST server send it a (set of) additional attribute(s) that the client is requested to supply in the subsequent enrollment (certificate signing) request.

3. Protocol Design and Layering

The following provides an expansion of Figure 1 describing how the layers are used. Each aspect is described in more detail in the sections below.

EST Layering:

Protocols and uses:

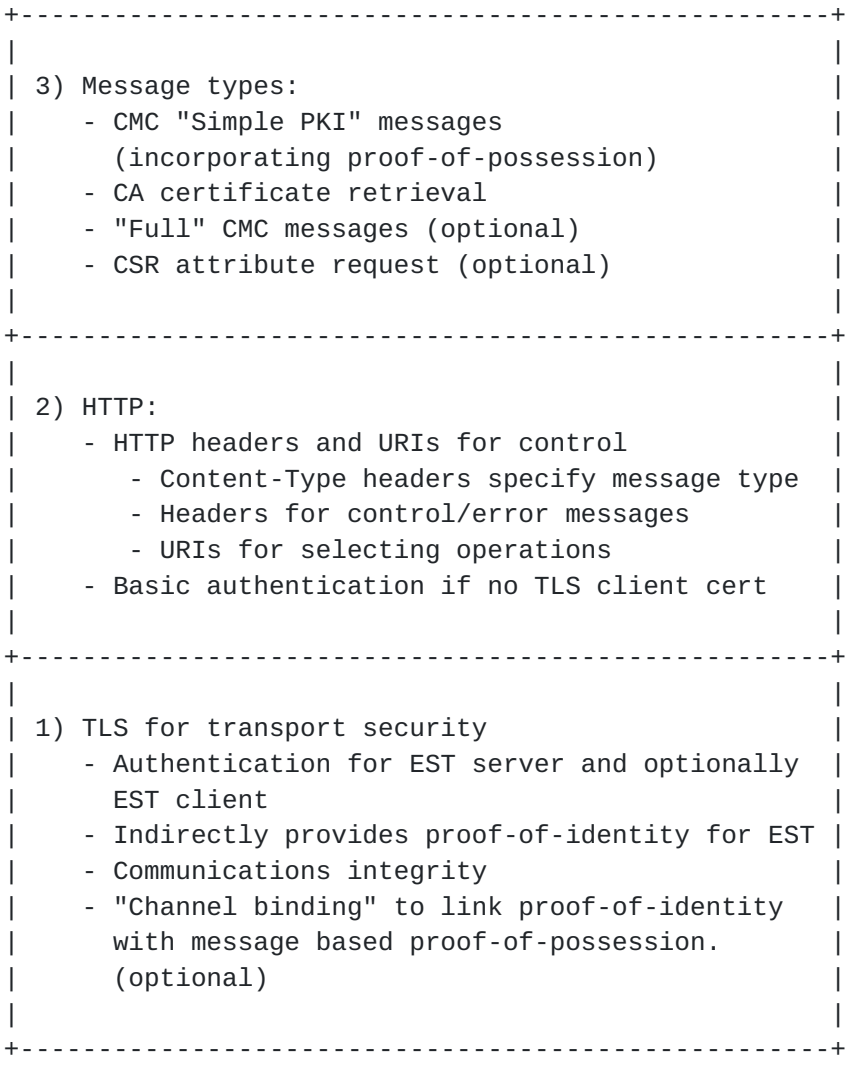

Figure 2

Specifying HTTPS as the secure transport for PKI enrollment messages introduces two 'layers' for communication of authentication and control messages during the protocol exchange: TLS and HTTP.

The TLS layer provides message authentication and integrity during transport. The proof-of-identity is supplied by either the certificate exchange during the TLS handshake or within the HTTP layer headers. The message type along with control/error messages are included in the HTTP headers.

The TLS and HTTP layer provided proof-of-identity means the CMC [RFC5272] Section 3.1 note that "the Simple PKI Request MUST NOT be

used if a proof-of-identity needs to be included" is not applicable and thus the "Simple PKI" message types are used.

The TLS layer certificate exchange provides a method for authorizing client enrollment requests using existing certificates. Such existing certificates may have been issued by the Certification Authority (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 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-identiy and proof-of-posession is described in Section 3.5.

This document also defines transport for the full CMC [RFC5272] specification compliant with CMC Transport Protocols [RFC5273].

During the protocol operations various different certificates can be used. The following table provides an informative overview. End entities MAY have one or more certificates of each type as is appropriate:

Certificates/Trust-anchors and their corresponding uses:

End Entity	Issuer	Use
EST server	The CA served by the EST server	To authenticate servers that have certs issued by the CA Section: 3.3.1.1.
EST server	An unrelated CA e.g., a Web site CA	To authenticate servers that have certs issued by Web site CAs Section: 3.3.1.1.
EST client Trust Anchor Database	Trust anchors for third party CAs e.g., a list of Web site CA root certs	EST clients can leverage a trust anchor database to authenticate EST servers using a configured URI Section: 3.3.1.1
EST client	An unrelated CA e.g., a device manufacturer	To authenticate clients that have not yet enrolled Section: 3.3.1.2
EST client	The CA served by the EST server	To authenticate clients that have already enrolled (for re-enroll or obtaining additional certs) Section: 3.3.1.2
EST client	The CA served by the EST server	Clients can obtain certs that can not be used for EST authentication (e.g., Key Encryption certs) Section: 4.4.1

Figure 3

3.1. Application Layer Design

An EST client SHOULD have its own client certificate suitable for TLS client authentication (e.g., the digitalSignature bit is set). The client certificate, if available, is used when authenticating to the EST server. This certificate MAY also be used by the client with other certificate consuming protocols. If a client does not have a certificate, then the client MUST have HTTP Basic or Digest

authentication credentials (see Section 3.2.3). HTTP authentication provides a bootstrap for clients that have not yet been issued an initial certificate. EST clients obtaining a certificates for other protocol purposes are RECOMMENDED to first obtain an appropropriate digitalSignature certificate for use when authenticating to the EST server.

The client also SHOULD also have a CA certificate that will be used to authenticate the EST server.

An EST client MUST be capable of generating and parsing simple CMC messages (see Section 4.4). Generating and parsing full CMC messages is optional (see Section 4.5). The client MUST also be able to request CA certificates from the EST server and parse the returned "bag" of certificates (see Section 4.3). Requesting CSR attributes and parsing the returned list of attributes is optional (see Section 4.7).

3.2. HTTP Layer Design

HTTP is used to transport EST requests and responses. Specific URIs are provisioned for handling each type of request 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, as detailed in Section Section 3.2.3. HTTP message types are used to convey EST requests and responses as specified in Figure 5.

3.2.1. HTTP headers for control

This document profiles the HTTP content-type header (as defined in [RFC2046], but see Figure 5 for specific values) to indicate the message type for EST messages and to specify EST control messages. The HTTP Status value is used to communicate success or failure of control messages. Support for the HTTP username/password methods is profiled for when a client does not have a suitable client certificate.

CMC does not provide specific messages for certificate renewal and certificate rekey. This profile defines the renewal and rekey behavior of both the client and server. It does so by specifying the HTTP control mechanisms employed by the client and server without requiring a new CMC message type.

Various media types as indicated in the HTTP content-type header are used to transport EST messages. Valid media types are specified in Section 3.2.4.

3.2.2. HTTP URIs for control

This profile supports four operations indicated by specific URIs:

Operations and their corresponding URIs:

Operation	Operation Path	Details
Distribution of CA certificates	/CACerts	Section 4.3
Enrollment of new clients	/simpleEnroll	Section 4.4
Re-Enrollment of existing clients	/simpleReEnroll	Section 4.4.1
Full CMC (optional)	/fullCMC	Section 4.5
Server-side Key Generation (optional)	/serverKeyGen	Section 4.6
Request CSR attributes (optional)	/CSRAttrs	Section 4.7

Figure 4

An HTTP base path common for all of an EST server's requests is defined in the form of an path-absolute ([RFC3986], section 3.3). The operation path (Figure 4 is appended to the base path to form the URI used with HTTP GET or POST to perform the desired EST operation.

An example:

With a base path of "/arbitrary/path" and an operation path of "/CACerts", the EST client would combine them to form an absolute path of "/arbitrary/path/CACerts". Thus, to retrieve the CA's certificates, the EST client would use the following HTTP request:

GET /arbitrary/path/CACerts HTTP/1.1

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

POST /arbitrary/path/simpleEnroll HTTP/1.1

The mechanisms by which the EST server interacts with an HTTPS server

to handle GET and POST operations at these URIs is outside the scope of this document. The use of distinct operation paths simplifies implementation for servers that do not perform client authentication when distributing "CACerts" responses.

EST clients are to be provided with the URL of the EST server and the base path. The means by which clients acquire the URL and base path are outside the scope of this document. Whether the URL and base path are provided securely determines the authorization scheme required to perform operations. (See Section 4.1.)

An EST server MAY provide additional, non-EST services on other URIs.

An EST server MAY use multiple base paths in order to provide service for multiple CAs. Each CA would use a distinct base path, but operations are otherwise the same as specified for an EST server operating on behalf of only one CA.

3.2.3. HTTP-Based Client Authentication

An EST server MAY fallback to using HTTP-based client authentication if TLS client authentication (Section 3.3.1.2) is not possible.

Basic and Digest authentication MUST only be performed over TLS 1.1 [RFC4346] (or later). 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 intended for deployments where password authentication is advantageous SHOULD support the Basic and Digest authentication mechanism. Servers MAY provide configuration mechanisms for administrators to enable Basic [RFC2616] and Digest [RFC2617] authentication methods.

Servers that support Basic and Digest authentication methods MAY reject requests using the HTTP defined WWW-Authenticate response-header ([RFC2616], Section 14.47). At that point the client SHOULD repeat 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 then the client MUST terminate the connection.

Clients MAY set the username to the empty string ("") if they wish to present a "one-time password" or "PIN" that is not associated with a username.

Support for HTTP-based client authentication has security ramifications as discussed in Section 6. The client MUST NOT respond to this request unless the client has authenticated the EST server

(as per Section 4.1).

3.2.4. Message types

This document uses existing media types for the messages as specified by Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP [RFC2585] and The application/pkcs10 Media Type [RFC5967] and CMC [RFC5272]. To support distribution of multiple application/pkcs7-mime's for the CA certificate chain the [RFC2046] multipart/mixed media type is used.

The message type is specified in the HTTP Content-Type header. The use herein is consistent with [RFC5273], with clarifications made concerning transfer encoding.

For reference the messages and their corresponding MIME and media types are:

Message type	Request type Response type Source(s) of types	Request section Response section
CA certificate request	N/A application/pkcs7-mime RFC 5751	Section 4.3 Section 4.3.1
Cert enroll/renew	application/pkcs10 application/pkcs7-mime RFC 5967, RFC 5751	Section 4.4/4.4.1 Section 4.4.2
Full CMC	application/pkcs7-mime application/pkcs7-mime RFC 5751	Section 4.5.1 Section 4.5.2
Server-side Key Generation	application/pkcs10 multipart/mixed (application/pkcs7-mime & application/pkcs8) RFC 5967, RFC 5751	Section 4.6.1 Section 4.6.2
Request CSR attributes	N/A application/csrattrs (Specified in this RFC)	Section 4.7.1 Section 4.7.2

Figure 5

3.3. TLS Layer Design

TLS provides communications security for the layers above it. Specifically, the integrity and authentication services it provides are leveraged to supply proof-of-identity and to allow authorization decisions to be made. TLS client authentication is the preferred method for identifying EST clients. In lieu of that, HTTP authentication protected by TLS encryption is also acceptable. Additionally, TLS channel binding information may be optionally inserted into a certificate request in order to provide the EST server with assurance that the authenticated TLS client entity has possession of the private key for the certificate being requested.

HTTP 1.1 [RFC2616] and above support persistent connections. As given 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 persistent HTTP connections and their use is out of scope of this specification.

3.3.1. TLS for transport security

HTTPS is defined in HTTP Over TLS [RFC2818] and is a specification of how HTTP messages are carried over TLS. HTTPS (e.g., HTTP over TLS) MUST be used. TLS 'session resumption' SHOULD be supported.

3.3.1.1. TLS-Based Server Authentication

The EST client MUST authenticate the EST server by validating the TLS server certificate the server presented during the TLS 1.1 [RFC4346] (or later) exchange-defined Server Certificate message or the client MUST independently validate the response contents. Validation is performed as given in [RFC5280] and [RFC6125].

There are multiple methods of validation depending on the current state of the client:

Method 1) If the client has a store of trust anchors, which may be in the form of certificates, for authenticating TLS connections the client MAY validate the TLS server certificate using the standard HTTPS logic of checking the server's identity as presented in the server's Certificate message against the URI provisioned for the EST server (see HTTP Over TLS [RFC2818], Section 3.1 "Server Identity" and [RFC6125]). This method makes it possible for clients with a store of trust anchors to securely obtain the CA certificate by leveraging the HTTPS security model. The EST server URI SHOULD be made available to the client in a secure fashion so that the client only obtains EST functions from a desired server.

Method 2) If the client already has one or more trust anchors associated with this EST server, the client MUST validate the EST server certificate using these trust anchors. The EST server URI MAY be made available to the client in an insecure fashion. The EST server certificate MUST contain the id-kp-cmcRA [CMC RFC5272bis] extended key usage extension.

Method 3) If the client does not yet have a trust anchor associated with this EST server then the client MAY provisionally accept the TLS connection, but the HTTP content data MUST be accepted manually as described in Section 4.3. HTTP authentication requests MUST NOT be responded to since the server is unauthenticated (only the content data is accepted manually).

Methods 1 and 2 are essentially validation as given in [RFC5280]. Method 1 is as described in [RFC6125], Section 6.6.1 "Match Found". Method 2 is described in [RFC6125] as "No Match Found, Pinned Certificate". Method 3 is described in [RFC6125], Section 6.6.4 as "Fallback" and describes the process of "pinning" the received certificate.

If one of these validation methods succeeds, the CA certificate(s) are stored and "pinned" for future use. If none of these validation methods succeeds the client MUST reject the EST server response and SHOULD log and/or inform the end user.

If Method 1 was used to authenticate the EST server then subsequent connections to the EST server also use Method 1. If Method 2 was used to authenticate the EST server then subsequent connections to the EST server also use Method 2. If Method 3 was used to manually authenticate the EST server then the EST client SHOULD "pin" the CA certificates received from a /CACerts (Section 4.3) operation and Method 2 is used for subsequent connections.

3.3.1.2. TLS-Based Client Authentication

Clients SHOULD support [RFC4346]-defined (or later) Certificate request (section 7.4.4). As required by [RFC4346], the client certificate needs to indicate support for digital signatures. The client SHOULD support this method in order to leverage /simpleReEnroll using client authentication by existing certificate. If a client does not support TLS client authentication, then it MUST support HTTP-based client authentication. (Section 3.2.3)

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

coresponding 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 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

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. This 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 CMC messages are used.

The client generating the request SHOULD obtain the tls-unique value as defined in Channel Bindings for TLS [RFC5929] from the TLS subsystem. The tls-unique value is encoded as specified in Section 4 of Base64 [RFC4648] and the resulting string is placed in the certification request challenge-password field. If tls-unique information is not embedded within the certification request the challenge-password field MUST be empty.

The tls-unique specification includes a synchronization issue as described in Channel Bindings for TLS [RFC5929] section 3.1. This problem is avoided for EST implementations. If the tls-unique value is used it MUST be from the first TLS handshake. EST client and servers use their tls-unique implementation specific synchronization methods to obtain this first tls-unique value.

If identity linking is used then TLS renegotiation MUST use "secure_renegotiation" [RFC5746] (thus maintaining the binding). Mandating secure renegotiation secures this method of avoiding the synchronization issues encountered when using the most recent tls-unique value (which is defined as the the value from the most recent TLS handshake).

The EST server MUST verify the tls-unique information embedded within the certification request and MUST reject requests with invalid tls-unique information. The EST server MAY be configured to accept requests from authenticated clients that do not include the tls-

unique information.

The tls-unique value is encoded into the certification request by the client but back-end infrastructure elements that process the request after the EST server might not have access to the initial TLS session. Such infrastructure elements validate the source of the certification request to determine if POP checks have already been performed. For example if the EST client authentication results in an authenticated client identity of an EST server RA that is known to independently verify the proof-of-possession then the back-end infrastructure does not need to perform proof-of-possession checks a second time. If the EST server forwards a request to a back-end process it SHOULD communicate the authentication results. This communication might use the CMC "RA POP Witness Control" in a CMC Full PKI Request message or other mechanisms which are out-of-scope of this document.

[[EDNOTE: A specific error code (TBD) is returned indicating this additional linkage might be useful. This would be similar to the "WWW-Authenticate response-header" control message. Alternatively simply rejecting the request with an informative text message would work in many use cases.]]

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 must make a determination if it will accept services from the EST server. Those determinations are described in the next two sections. Assuming that both sides of the exchange are authorized, then the actual operations are as described in the sections following.

4.1. Server Authorization

The client MUST check the EST server authorization before accepting the server's response. The presented certificate MUST be an end-entity certificate such as a CMC Registration Authority (RA) certificate.

There are multiple methods for checking authorization corresponding to the method of server authentication used (these authorization methods align with the authentication methods described in Section 3.3.1.1):

Method 1) If the client authenticated the EST server using the client's TLS trust anchors store, then the client MUST have obtained the EST server's URI in a secure fashion. The client MUST check the URI "against the server's identity as presented in the server's Certificate message" (Section 3.1 "Server Identity" [RFC2818] and [RFC6125]). The securely configured URI provides the authorization statement and the server's authenticated identity confirms it is the authorized server.

Method 2) If the previous check fails or is not applicable, or if the EST server's URI was made available to the client in an insecure fashion, then the EST server certificate MUST contain the id-kp-cmcRA [CMC RFC5272bis] extended key usage extension. The client MUST further verify the server's authorization by checking that the [RFC5280]-defined certificate policy extension sequence contains the 'RA Authorization' policy OID. The RA Authorization policy OID is defined as: id-cmc [[EDNOTE: TBD, perhaps 35]]. The RA Authorization policy information MUST NOT contain any optional qualifiers.

Method 3) If fallback logic was invoked to accept the certificate manually, then that authentication implies authorization of the EST server.

4.2. Client Authorization

When the EST server receives a CMC Simple PKI Request or rekey/renew message, the decision to issue a certificates is always a matter of local policy. Thus the CA can use any data it wishes in making that determination. The EST protocol exchange provides the EST server access to the TLS client certificate in addition to any HTTP user authentication credentials to help in that determination. The communication channel between the TLS server implementation and the EST software implementation is out-of-scope of this document.

If the client authentication is incomplete (for example if the client certificate is self-signed or issued by an unknown PKI or if the client offered an unknown username/password during HTTP authentication) the server MUST extract the certificate request for manual authorization by the administrator.

4.3. Distribution of CA certificates

The EST Client MAY request trust anchor information of the CA (in the form of certificates) by sending an HTTPS GET message to the EST server with an operations path of "/CACerts". Clients SHOULD request an up-to-date response before stored information has expired in order to maintain continuity of trust.

The EST server SHOULD 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 check the server's authorization as given in Section 4.1. If the TLS authentication and authorization is not successful then the client MAY continue the TLS handshake to completion and proceed with the /CACerts request. If the EST client continues with an unauthenticated connection the EST client MUST extract the CA certificate from the response (Section 4.3.1) and engage the end-user to authorize the CA certificate using out-of-band pre-configuration data such as a CA certificate "fingerprint" (e.g., a SHA-1, SHA-256, SHA-512, or MD5 hash on the whole CA certificate). In this case it is incumbent on the end user to properly verify the fingerprint or to provide valid out-of-band data necessary to verify the fingerprint.

4.3.1. Distribution of CA certificates response

The EST server MUST respond to the client HTTPS GET message with CA trust anchor information, in the form of certificates within the CMC Simple PKI Response. The response is conveyed within an HTTP response.

The EST server MUST include the current CA certificate in the response. The EST server MUST include any additional certificates the client would need to build a chain to the root certificate. For example if the EST server is configured to use a subordinate CA when signing new client requests then the appropriate subordinate CA certificates to chain to the root must be included in the response.

Additional certificates MAY be included. If support for the CMP root certificate update mechanism is provided by the CA then the server MUST include the three "Root CA Key Update" certificates OldWithOld, OldWithNew, and NewWithOld. These are defined in Section 4.4 of CMP [RFC4210].

The client can always find the current self-signed CA certificate by examining the certificates received. The NewWithNew certificate is self-signed and has the latest NotAfter date.

The NewWithNew certificate is the certificate that is extracted and authorized using out-of-band information as described in Section 4.3. When out-of-band validation occurs each of the other three certificates MUST be validated using normal [RFC5280] certificate path validation (using the NewWithNew certificate as the trust anchor) before they can be used to build certificate paths during peer certificate validation.

The response format is the CMC Simple PKI Response as defined in [RFC5272]. The HTTP content-type of "application/pkcs7-mime" MUST be specified. The CMC Simple PKI response is Base64 encoded and sandwiched between PEM headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBFMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of Base64 encoding of CMC Simple PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVEryb1e4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PKCS7-----
```

4.4. Simple Enrollment of Clients

The EST client MAY request a certificate from the EST server by HTTPS POSTing using the operation path value of "/simpleEnroll".

When HTTPS POSTing to the 'SimpleEnroll' location the client MUST include a CMC Simple PKI Request as specified in CMC Section 3.1 (i.e., a PKCS#10 Certification Request). Consistent with [RFC6403] the certification request "signature MUST be generated using the private key corresponding to the public key in the CertificationRequestInfo, for both signature and key establishment certification requests". The signature provides proof-of-possession of the private key to the EST server.

The HTTP content-type of "application/pkcs10" MUST be specified. The format of the request is as specified in Section 6.4 of [RFC4945].

The server MUST check client authorization as specified in Section 4.2. The EST server MUST check the tls-unique value as described in Section 3.5 but depending on policy MAY accept a request without the encoded tls-unique value. The EST server applies whatever authorization or policy logic it chooses in determining if the certificate should be issued.

The optional client signature certificate MAY be an existing certificate issued by the CA the EST server is providing services for or it MAY be from any other PKI the EST server indicated as acceptable during the TLS handshake.

The client MAY request an additional certificate even when using an existing certificate in the TLS client authentication. For example the client can use an existing signature certificate to request a key encryption certificate.

The client MUST authenticate the EST server as specified in Section 3.3.1.1.

4.4.1. Simple Re-Enrollment of Clients

The EST client MAY request renew/rekey of its certificate from the EST server by HTTPS POSTing using the operation path value of "/simpleReEnroll'.

The certificate request is the same format as for the "simpleEnroll" path extension with the same HTTP content-type.

The server MUST check client authorization as specified in Section 4.2. The EST server MUST check the tls-unique value as described in Section 3.5 but depending on policy MAY accept a request without the encoded tls-unique value. The server applies whatever authorization or policy logic it chooses in determining if the certificate should be renewed/rekeyed. The optional client signature certificate MAY be an existing certificate issued by the CA the EST server is providing services for or it MAY be from any other PKI the EST server indicated as acceptable during the TLS handshake. When attempting to renew or rekey the client SHOULD use an existing certificate for TLS client authentication (Section 3.3.1.2). The certificate being re-enrolled MAY be different than the certificate used for EST client authentication.

The EST server MUST handle enrollment requests submitted to the "simpleReEnroll" URI as a renewal or rekey request. (This explicit method of indicating a re-enroll request is an alternative to the /fullCMC method specified in Section 2 of [RFC5272] wherein the "renewal and rekey requests look the same as any certification request, except that the identity proof is supplied by existing certificates from a trusted CA").

The request Subject/SubjectAltName field(s) MUST contain the identity of the certificate being re-enrolled. The ChangeSubjectName attribute, as defined in [RFC6402] MAY be included in the certificate request. The EST server MUST verify that that authenticated client is authorized to perform the inferred re-enroll operation.

If the public key information in the certification request is the same as the currently issued 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. The specifics of these operations are out of scope of this profile.

The client MUST authenticate the EST server as specified in Section 3.3.1.1. The EST client is RECOMMENDED to have obtained the current CA certificates using Section 4.3 to ensure it can validate the EST server certificate.

4.4.2. Simple Enroll and Re-Enroll Response

If the enrollment is successful the server response MUST have an HTTP 200 response code with a content-type of "application/pkcs7-mime". The response data is a degenerate certs- only CMC Simple PKI Response containing only the certificate issued. The CMC Simple PKI response is Base64 encoded and sandwiched between PEM headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBFMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of Base64 encoding of CMC Simple PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVEryb1e4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PKCS7-----
```

When rejecting a request the server MUST specify either an HTTP 4xx/ 401 error, or an HTTP 5xx error. A CMC PKI Response with an HTTP content-type of "application/pkcs7-mime" MAY be included in the response data for any error response. If the content-type is not set the response data MUST be a plain text human-readable error message. A client MAY elect not to parse a CMC error response in favor of a generic error message.

If the server responds with an HTTP 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 and 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.

If the EST client has not obtained the current CA certificates using Section 4.3 then it may not be able to validate the certificate received.

4.5. Full CMC

The EST client MAY request a certificate from the EST server by HTTPS POSTing using the operation path value of "/fullCMC".

The client MUST authenticate the server as specified in Server

Authentication (Section 3.3.1.1), if method 3 is used, then the Publish Trust Anchors control within the HTTP content must be accepted manually as noted in Section 4.3. While use of TLS is not optional within EST, since a full CMC message inately provides security, a TLS NULL cipher suite may be used while making this request.

The server SHOULD authenticate the client as specified in Section 3.3.1. The server MAY depend on CMC client authentication methods instead.

4.5.1. Full CMC Request

When HTTPS POSTing to the "fullCMC" location the client MUST include a valid CMC message. The HTTP content-type MUST be set to "application/pkcs7-mime" as specified in [RFC5273].

4.5.2. 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 have an HTTP 200 response code with a content-type of "application/pkcs7-mime" as specified in [RFC5273]. The response data includes either the CMC Simple PKI Response or the CMC Full PKI Response.

When rejecting a request the server MAY specify either an HTTP 4xx/ 401 error or an HTTP 5xx error. A CMC response with content-type of "application/pkcs7-mime" MUST be included in the response data for any error response. The client MUST parse the CMC response to determine the current status.

All other return codes are handled as specified in Section 4.4.2 or HTTP [RFC2616].

The CMC PKI response is Base64 encoded and sandwiched between PEM headers:

```
-----BEGIN PKCS7-----
MIIBhDCB7gIBADBFMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of Base64 encoding of CMC Full PKI Response
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVEryb1e4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PKCS7-----
```

4.6. Server-side Key Generation

[[EDNOTE: This section is references [draft-ietf-pkix-cmc-serverkeygeneration-00] which has not yet been published.]]

The EST client MAY request a "private" key and associated certificate from the EST server by HTTPS POSTING using the operation path value of "/serverKeyGen".

The client MUST authenticate the server as specified in Section 3.3.1.1. The EST client is RECOMMENDED to have obtained the current CA certificates using Section 4.3 to ensure it can validate the EST server certificate.

The EST server MUST authenticate the client as specified in Section 3.3.1. The EST server applies whatever authorization or policy logic it chooses to determine if the "private" key and certificate should be distributed. The server SHOULD use TLS-Based Client Authentication for authorization purposes. The server SHOULD respond to repeated requests from the same client with the same "private" key and certificate but MAY respond with a renewed or rekeyed "private" key and certificate. Clients that wish multiple "private" keys and certificates MUST specify a keyUsage in the certificate request which the server will use to intuit the type of key to be generated.

Proper random number and key generation and storage is a server implementation responsibility. The keypair and certificate are transfered over the TLS session; the EST server MUST verify that the current ciphersuite is acceptable for securing the key data.

4.6.1. Server-side Key Generation Request

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

The public key values of the certificate request and the request signature MUST be ignored by the server.

4.6.2. 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. The first part is the "private" key data and the second part is the certificate data.

The first submessage is an "application/pkcs8" consisting of the

Base64 encoded DER-encoded PrivatekeyInfo sadwiched between the PEM headers as described in [RFC5958]:

```
-----BEGIN PRIVATE KEY-----
MIIBhDCB7gIBADBFMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEh
Simplified example of Base64 encoding of DER-encoded PrivateKeyInfo
ED8rf3UDF6HjloiV3jBnpetx4JjZH/BlmD9HMqofVEryb1e4iZgMUvuIgwEjQwpD
8J4OhHvLh1o=
-----END PRIVATE KEY-----
```

The second submessage is an "application/pkcs7-mime" and exactly matches the certificate response to /simpleEnroll. The server response MUST use the same SubjectPublicKeyInfo as requested or the request MUST be denied.

When rejecting a request the server MUST specify either an HTTP 4xx/401 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.

4.7. CSR Attributes

The CA MAY want to include client-provided attributes in certificates that it issues and some of these attributes may describe information that is not available to the CA. For this reason, the EST client MAY request a set of attributes from the EST server to include in its certification request.

4.7.1. CSR Attributes Request

The EST Client MAY request 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". Clients SHOULD request such a list if they have have no a priori knowledge of what attributes are desired by the CA in an enrollment request or when dictated by policy.

4.7.2. CSR Attributes Response

The server MUST reply to the client's HTTPS GET message with a (set of) attribute(s). Responses to attribute request messages MUST be encoded as content type "application/csrattrs" and conveyed within an HTTP response.

The syntax for application/csrattrs body is as follows:

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

A robust application SHOULD output Distinguished Encoding Rules (DER)

([X.690]) but MAY use Basic Encoding Rules (BER) ([X.680]). Data produced by DER or BER is 8-bit. When the transport for the application/csrattrs is limited to 7-bit data, a suitable transfer encoding MUST be applied in MIME-compatible transports. The base64 encoding (section 4 of [RFC4648]) SHOULD be used with application/csrattrs, although any 7-bit transfer encoding may work.

Servers include zero or more object identifiers that they wish the client to include in their certification request. When the server encodes csrattrs as an empty SEQUENCE OF it means that the server has no attributes it wants in client certification requests.

For example, if a CA wishes to have a certification request contain the MAC address [RFC2397] of a device and the pseudonym [X.520] and friendly name [RFC2925] of the holder of the private analog to the public key in the certification request, it takes the following object identifiers:

o macAddress: 1.3.6.1.1.1.1.22

o pseudonym: 2.5.4.65

o friendlyName: 1.2.840.113549.1.9.20

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

```
30 19 06 07 2b 06 01 01 01 01 16 06 03 55 04 41 06 09 2a 86 48 86
f7 0d 01 09 14
```

and then base64 encodes the resulting ASN.1 SEQUENCE to produce:

```
MBkGBysGAQEBARYGA1UEQQYJKoZIhvcNAQkU
```

The resulting response would look like this:

```
Content-Type: application/csrattrs; name=attributes
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=attributes

MBkGBysGAQEBARYGA1UEQQYJKoZIhvcNAQkU
```

5. IANA Considerations

(This section is incomplete)

The following aspects should be registered with IANA Considerations:

The RA Authorization certificate policy extension OID as discussed in Section 4.1 requires registration with IANA.

[[EDNOTE: The URLs specified in Section 1 probably do not need to be registered with IANA.]]

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

6. Security Considerations

(This section is incomplete)

"Badges? We ain't got no badges. We don't need no badges! I don't have to show you any stinkin' badges!" -- The Treasure of the Sierra Madre.

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 the proof-of-possession to the TLS proof-of-identity.

As given in Section 3.3.1.2 clients use an existing certificate for TLS client authentication. If a certificate with appropriate key usage is not available the client MAY generate one. If a self-signed certificate with appropriate key usage is used the server SHOULD require HTTP-based client authentication according to server policy as described in Section 3.3.1.2 and Section 4.2. The server MAY fallback on manual authorization by the server administrator.

Clients authenticate EST servers by means of TLS authentication. If a client does not possess a root certificate suitable for validating an EST server certificate, it MAY rely upon other trusted root certificates it has (such as those found in its HTTPS store). The client then is able to retrieve additional root certificates as given in Section 4.3. Alternatively, a server certificate MAY be authenticated manually as specified in Section 3.3.1.1 #3.

As noted in Section 3.3.1.1 servers use an existing certificate for TLS server authentication. When the server certificate is issued by a mutually trusted PKI hierarchy, validation proceeds as specified in Section 4.1. In this situation the client has validated the server as being a valid responder for the URI configured but can not directly verify that the responder is authorized as an RA within the to-be-enrolled PKI hierarchy. A client may thus be enticed to expose username/password or certificate enrollment requests to an unauthorized server (if the server presents a valid HTTPS certificate

for an erroneous URL that the client has been tricked into using). Proof-of-identity and Proof-of-possession checks by the CA prevent an illegitimate RA from leveraging such misconfigured clients to act as a man-in-the-middle during client authenticated operations but it is possible for such illegitimate RAs to send the client doctored messages or erroneous CA certificate lists. If the illegitimate RA has successfully phished a username/password or PIN from the client it might try to use these values to enroll its own keypair with the real PKI hierarchy. EST servers identified with an externally issued server certificate SHOULD require HTTPS-based client authentication (Section 3.3.1.2). Similarly EST clients SHOULD use an existing client certificate to identify themselves and otherwise prevent "private data" (obviously including passwords but also including private identity information) from being exposed during the enrollment exchange a weak server authorization method is used.

Section 3.2.3 allows clients to optionally authenticate using HTTP-based authentication in place of TLS-based authentication. HTTP-based authentication MUST NOT take place unless performed over a TLS-protected link.

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 and servers are NOT RECOMMENDED to support this operation by default. Clients are NOT RECOMMENDED to request this service unless there is a compelling operational benefit such as the use of [BGPsec RPKI].

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.

[[EDNOTE: need final reference for BGPsec RPKI]]

Support for Basic authentication as specified in HTTP [RFC2617] allows the server access to the client's cleartext password. This provides integration with 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 concerns but EST clients are RECOMMENDED to use such credentials only once to obtain an appropriate client certificate to be used during future interactions with the EST server.

7. References

7.1. Normative References

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

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

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

[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.

[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.

[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", BCP 13, RFC 4288, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, 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.

[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.

[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.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, February 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.

[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>.

7.2. 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.

[RFC2925] White, K., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", RFC 2925, September 2000.

[RFC6403] Zieglar, L., Turner, S., and M. Peck, "Suite B Profile of Certificate Management over CMS", RFC 6403, November 2011.

[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. Server Discovery

(informative)

Clients can use DNS-SD or similar discovery algorithms to determine the EST server URI. In such cases it is expected that method 2 (Section 3.3.1.1) be used during server authentication because the first method is insecure if the discovery mechanism is insecure.

If the user interaction in the third method is acceptable it is expected that the user would also supply the URI instead of using a discovery protocol.

Appendix B. External TLS concentrator

(informative)

In some deployments it may be beneficial to use a TLS concentrator to offload the TLS processing from the server.

The TLS server should not reject the connection based on PKIX validation of the client certificate. Instead the client certificate is passed to the EST server layer for verification and authorization. This allows support of external TLS concentrators that might provide an independent TLS implementation.

The TLS concentrator does validate the TLS Section 7.4.8 'Certificate Verify'.

In such a deployment the TLS client authentication result must be

forwarded to the EST server layer. For example a TLS concentrator might insert the client certificate into the HTTP header (first removing any existing client certificates, possibly inserted by a nefarious client, from the HTTP headers) before forwarding the HTTP connection to the EST server.

The EST server MUST be specifically configured by the administrator to accept this mechanism.

Appendix C. CGI Server implementation

(informative)

In some deployments it may be beneficial to use a HTTPS server that runs the EST server as a CGI application.

The HTTPS server should not reject the connection based on PKIX validation of the client certificate. Instead the client certificate is passed to the EST server layer for verification and authorization. This allows support of external HTTPS servers that might provide an independent TLS implementation.

In such a deployment the TLS client authentication result must be forwarded to the EST server layer. For example an HTTPS server might insert the client certificate into the environment variables before forwarding the HTTP data to the EST server.

Appendix D. Operational Scenario Example Messages

(informative)

This section expands on the Operational Scenario Overviews by providing detailed examples of the messages at each TLS layer. Figures are informative sections of TLSv1

D.1. Obtaining CA Certificates

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

During the intial 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
kqhkiG9w0BBwGgggLkMIIC.4DCCAcigAwIBAgIJAOjxMZcXhE5wMA0GCSqGSIb3D
QEBBQUAMBcxFTATBgNVBAMT.DGVzdEV4YW1wbGVDQTAeFw0xMjA3MDQxODM5Mjda
Fw0xMzA3MDQxODM5MjdaMBcx.FTATBgNVBAMTDGVzdEV4YW1wbGVDQTCCASIwDQY
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.BgNVHQ4EFgQUhH9KxW5TsjkgL7kg2kxJ
yy5tD/MwDQYJKoZIhvcNAQEFBQADggEB.AD+vydZo292XFb2vXojdKD57Gv4tKVm
hvXRdVInntzkY/0AyFCfHJ4BwndgtMh4t.rvBD8+8dL+W3jfPjcSCcUQ/JEnFuMn
b5+kivLeqOnUshETasFPBz2Xq4C1sHDno9.CWOcsjPPw08Tn4dSrzDBSq1NdXB2z
9NOpaVnbpb01qQGhXSOaEvcbZcDuGiW7Di3.gV++remokuPph/s6XoZffzc7ZVzf
Job6tS4RwNz01sutPybXiRWivOz7+QeCOT87.nTGlkQH/+RImUyJ2jefjAW/GDFT
Pzek6cZnabAtsg32n0Pv0j0/1RTNSdYGxPIVA.2f9fhMqMz+vm3w4CFNkGZnOhAD
EA.-----END PKCS7-----.
```

D.2. Previously Installed Signature Certificate

The following is an example of a valid /simpleEnroll exchange. During this exchange the EST client uses an existing certificate issued by a trusted 3rd party PKI 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 CA the EST server provides services for ("estExampleCA") and it includes the distinguished name of a trusted 3rd 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/x-est-pkcs10
Content-Length: 952

=> Send data, 952 bytes (0x3b8)
-----BEGIN CERTIFICATE REQUEST-----.MIIChjCCAW4CAQAwQTElMCMGA1UE
AxMccmVxIGJ5IGNsaWVudCBpbiBkZW1vIHN0.ZXAgNjEYMBYGA1UEBRMPUElEOld
pZGdldCBTTjo2MIIBIjANBgkqhkiG9w0BAQEF.AAOCAQ8AMIIBCgKCAQEAwhYyI+
aYezyx+kW0GVUbMKLf2BUd8BgGykkIJYxms6SH.Bv5S4ktcpYbEpR9iCmp96vK6a
Ar57ArZtMmi0Y6eLX4c+njJnYhUeTivnfyfMM5d.hNVwyzKbJagm5f+RLTMfp0y0
ykqrfZ1hFhcNrRzF6mJeaORTHBehMdu8RXcbmy5R.s+vjnUC4Fe3/oLHtXePyYv1
qqlkk0XDrw/+lx0y4Px5tiyb84iPnQOXjG2tuStM+.iEvfpNAnwU0+3GDjl3sjx0
+gTKvblp6Diw9NSaqIAKupcgWsA0JlyYkgPiJnXFKL.vy6rXoOyx3wAbGKLrKCxT
l+RH3oNXf3UCH70aD758QIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBADwpafWU
BsOJ2g2oyHQ7Ksw6MwvimjhB7GhjweCcceTSLInUMk10.4E0TfNqaWcoQengMVZr
IcbOb+sa69BWNB/WYIULfEtJIV23/g3n/y3JltMNw/q+R.200t0bNAViijHQHmlF
6dt93tkRrTzXnhV70Ijnff08G7P9HfnXQH4Eiv3zOB6Pak.JoL7QlWQ+w5vHpPo6
WGH5n2iE+Ql76F0HykGeqaR402+ae0WlGLHEvcN9wiFQVKh.KUHteU10SEPijlqf
QW+hciLleX2CwuZY5MqKb4qqyDTs4HSQCBCl8jR2cXsGDuN4.PcMPp+9A1/UPuGD
jhwPt/K3y6aV8zUEh8Ws=.-----END CERTIFICATE REQUEST-----.
```

The EST server uses the trusted 3rd 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-----.MIIDUQYJKoZIhvcNAQcCoIIDQjCCAz4CAQExADALBg
kqhkiG9w0BBwGgggMkMIID.IDCCAgigAwIBAgIBBjANBgkqhkiG9w0BAQUFADAXM
RUwEwYDVQQDEwxlc3RFeGFt.cGxlQ0EwHhcNMTIwNzA0MTgzOTM3WhcNMTMwNzA0
MTgzOTM3WjBBMSUwIwYDVQQD.ExxyZXEgYnkgY2xpZW50IGluIGRlbW8gc3RlcCA
2MRgwFgYDVQQFEw9QSUQ6V2lk.Z2V0IFNOOjYwggEiMA0GCSqGSIb3DQEBAQUAA4
IBDwAwggEKAoIBAQDCFjIj5ph7.PLH6RbQZVRswot/YFR3wGAbKSQgljGazpIcG/
lLiS1ylhsSlH2IKan3q8rpoCvns.Ctm0yaLRjp4tfhz6eMmdiFR5OK+d/J8wzl2E
1XDLMpslqCbl/5EtMx+nTLTKSqt9.nWEWFw2tHMXqYl5o5FMcF6Ex27xFdxubLlG
z6+OdQLgV7f+gse1d4/Ji/WqqWSTR.cOvD/6XHTLg/Hm2LJvziI+dA5eMba25K0z
6IS9+k0CfBTT7cYOOXeyPHT6BMq9uW.noOLD01JqogAq6lyBawDQmXJiSA+ImdcU
ou/Lqteg7LHfABsYousoLFOX5Efeg1d./dQIfvRoPvnxAgMBAAGjTTBLMAkGA1Ud
EwQCMAAwHQYDVR0OBBYEFJv4oLLeNxNK.OMmQDDujyNR+zaVPMB8GA1UdIwQYMBa
AFIR/SsVuU7I5IC+5INpMScsubQ/zMA0G.CSqGSIb3DQEBBQUAA4IBAQCMdomfdR
9vi4VUYdF+eym7F8qVUG/1jtjfaxmrzKeZ.7LQ1F758RtwG9CDu2GPHNPjjeM+DJ
RQZN999eLs3Qd/DIJCNimaqdDqmkeBFC5hq.LZOxbKhSmhlR7YKjIZuyI299rOaI
W54ULyz8k0zw6R1/0lMJTsDFGJM+9yDeaARE.n3vtKnUDGHsVU3fYpDENaqUunoU
MZfuEdejfHhU7lVbJI1oSJbnRwBFkPr/RQ3/5.FymcrBD9RpAM5MsQIn0BONil/o
JM+LjOJqyZLbBxz6P3w/OiJGYJNfFT8YudLfjZ.LDX8A8FFcReapNELC4QxE4OrA
hN3sQUT2O7ndIsit4kJoQAxAA==.-----END PKCS7-----.
```

D.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 intial 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 OpenS
SL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 127.0.0.1:8085
Accept: */*
Content-Type: application/x-est-pkcs10
Content-Length: 952

=> Send data, 952 bytes (0x3b8)
-----BEGIN CERTIFICATE REQUEST-----.MIIChjCCAW4CAQAwQTElMCMGA1UE
AxMccmVxIGJ5IGNsaWVudCBpbiBkZW1vIHN0.ZXAgMjEYMBYGA1UEBRMPUElEOld
pZGdldCBTTjoyMIIBIjANBgkqhkiG9w0BAQEF.AAOCAQ8AMIIBCgKCAQEAz9lXz9
MowulOx0W5v1k7GKlsNy7mAgmkz/wZDImBDXez.QZCb8lrO8iTD3tI0NH2xpkY3b
uqFjdtQTzCmANLyNWtR1sC5GjN/EM1JSCrO/zZM.ig835RXJTP878N/jNW7EzSxb
/zK5OzKJoRbZ4HgZm4NDapMfMcB4jqBdPxoPAqeR.+KTkv1+9m1vvsdKIs5Hm4Sp
O2WolHPw5BCXdu5zleb6ACih7Zpd2cpHFz6ZHC0G1.Of+F//0BzkFSqWsmUomyJy
WCfLCuX9grs1CNlLxw0gcMprdTxLxjc18z03ZmBCq0.qq5/mUK/tv9R2k8+WuP3a
kzTUIkeHtcp6FVFl3D+TwIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBAJH7Etuy
B/oQgQeals08mD2U31FfQ/uYqjNxzZpZJSzVLGMASv9a.pNzaWdfqPdIs+ZZ+gAQ
QkVcXjdbqY3pAf/EeWk+KnuAUjOIPKu3ZBPVbWbXu/Ie7.F1ekQ7TLkFNkHSxHRu
2/bPIByBLRVfWNVXd3wPq+QxqMqgIjBGaTJM5kuHndYFGj.Xdf4rlGRPyOOwG/Xf
QrKBB3tzpbJCy+cwOUAJFPOTO+86RUjf9Wh+yoM182vlg8O.FyEaaA/PMpl3aEcT
BlRZmPx4e7FLwGIhbgE7/6K0nF99xdGd7JYPHasbcWszxD0Z.oPYm+44g0gOnhlj
OWpRiKXcnngSSutRILaw=.-----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/x-est-pkcs10
   Content-Length: 952

   => Send data, 952 bytes (0x3b8)
   -----BEGIN CERTIFICATE REQUEST-----.MIIChjCCAW4CAQAwQTElMCMGA1UE
   AxMccmVxIGJ5IGNsaWVudCBpbiBkZW1vIHN0.ZXAgMjEYMBYGA1UEBRMPUElEOld
   pZGdldCBTTjoyMIIBIjANBgkqhkiG9w0BAQEF.AAOCAQ8AMIIBCgKCAQEAz9lXz9
   MowulOx0W5v1k7GKlsNy7mAgmkz/wZDImBDXez.QZCb8lrO8iTD3tI0NH2xpkY3b
   uqFjdtQTzCmANLyNWtR1sC5GjN/EM1JSCrO/zZM.ig835RXJTP878N/jNW7EzSxb
   /zK5OzKJoRbZ4HgZm4NDapMfMcB4jqBdPxoPAqeR.+KTkv1+9m1vvsdKIs5Hm4Sp
   O2WolHPw5BCXdu5zleb6ACih7Zpd2cpHFz6ZHC0G1.Of+F//0BzkFSqWsmUomyJy
   WCfLCuX9grs1CNlLxw0gcMprdTxLxjc18z03ZmBCq0.qq5/mUK/tv9R2k8+WuP3a
   kzTUIkeHtcp6FVFl3D+TwIDAQABoAAwDQYJKoZIhvcN.AQEFBQADggEBAJH7Etuy
   B/oQgQeals08mD2U31FfQ/uYqjNxzZpZJSzVLGMASv9a.pNzaWdfqPdIs+ZZ+gAQ
   QkVcXjdbqY3pAf/EeWk+KnuAUjOIPKu3ZBPVbWbXu/Ie7.F1ekQ7TLkFNkHSxHRu
   2/bPIByBLRVfWNVXd3wPq+QxqMqgIjBGaTJM5kuHndYFGj.Xdf4rlGRPyOOwG/Xf
   QrKBB3tzpbJCy+cwOUAJFPOTO+86RUjf9Wh+yoM182vlg8O.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
kqhkiG9w0BBwGgggMkMIID.IDCCAgigAwIBAgIBAjANBgkqhkiG9w0BAQUFADAXM
RUwEwYDVQQDEwxlc3RFeGFt.cGxlQ0EwHhcNMTIwNzA0MTgzOTM0WhcNMTMwNzA0
MTgzOTM0WjBBMSUwIwYDVQQD.ExxyZXEgYnkgY2xpZW50IGluIGRlbW8gc3RlcCA
yMRgwFgYDVQQFEw9QSUQ6V2lk.Z2V0IFNOOjIwggEiMA0GCSqGSIb3DQEBAQUAA4
IBDwAwggEKAoIBAQDP2VfP0yjC.6U7HRbm/WTsYqWw3LuYCCaTP/BkMiYENd7NBk
JvyWs7yJMPe0jQ0fbGmRjdu6oWN.21BPMKYA0vI1a1HWwLkaM38QzUlIKs7/NkyK
DzflFclM/zvw3+M1bsTNLFv/Mrk7.MomhFtngeBmbg0Nqkx8xwHiOoF0/Gg8Cp5H
4pOS/X72bW++x0oizkebhKk7ZaiUc./DkEJd27nOV5voAKKHtml3ZykcXPpkcLQb
U5/4X//QHOQVKpayZSibInJYJ8sK5f.2CuzUI2UvHDSBwymt1PEvGNzXzPTdmYEK
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
AC2aHt9VPRViov4RRTvoQAxAA==.-----END PKCS7-----.
```

D.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 CA the EST server provides services for.

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 3rd 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.

D.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/x-est-pkcs10
   Content-Length: 968

   => Send data, 968 bytes (0x3c8)
   -----BEGIN CERTIFICATE REQUEST-----.MIICkzCCAXsCAQAwTjEyMDAGA1UE
   AxMpc2VydmVyS2V5R2VuIHJlcSBieSBjbGll.bnQgaW4gZGVtbyBzdGVwIDUxGDA
   WBgNVBAUTD1BJRDpXaWRnZXQgU046NTCCASIw.DQYJKoZIhvcNAQEBBQADggEPAD
   CCAQoCggEBAMnlUlq0ag/fDAVhLgrXEAD6WtZw.Y2rVGev5saWirer2n0OzghB59
   uJByxPo0DYBYqZRuoRF0FTL1ZZTMaZxivge0ecA.ZcoR46jwSBoceMT1jkwFyAER
   t9Q2EwdnJLIPo/Ib2PLJNb4Jo8NNKmxtg55BgIVi.vkIB+rMtLeYRUVL0RUaBAqX
   FmtXRDceVFIEY24iUQw6vESGJKpArht592aT8lyaP.24bZovuG19dd5xtTX3j37K
   x49SlkUvLSpD6ZavIFAZn7Yv19LBKHvRIemybUo294.QeLb/VYP1O+EAthV/igiX
   1DHqlUZCZp5SdyUXUwZPatFboNwEVR0R3MJwVECAwEA.AaAAMA0GCSqGSIb3DQEB
   BQUAA4IBAQAqhHezK5/tvbXleHO/aTBVYO9l414NM+WA.wJcnS2UaJYScPBqlYK/
   gij+dqAtFE+5ukAj56t7HnooI4EFo9r8jqCHewx7iLZYh.JDxo4hWOsAvHV+Iziy
   jkhJNdHBIqGM7Gd5f/2VJLEPQPmwNOL5P+2O4eQC/QeEYc.bAmfhOS8b/ZH09/9T
   PeaeQpjspjOui/100OuLE8KvU3FM0sXMYt1Va0A0jxzl+5k.EiEJo+ltXsQwdP0H
   csoTNBN+j3K18omJQS0e91X8v0xkMWYhUtonXD0YZ6SO/B9c.AE6GTADHA/xpSvA
   cqlWa+FHxjwEMXdmViHvMUywo31fDZ/TUvCPX.-----END CERTIFICATE REQUE
   ST-----.

   After processing the request 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
   BgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC0781l7tri0yii.Mb9ZZYch8ze
   izXrjMPF/Rxoz2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSM.tzxn4l+9tI
   sVDkAe4FyzN0hLd/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImDV3blv.aMUKDSJhV
   bQ+z/G1W1TRx3iWi5CMHYb+1pJXPTJz/GuWr/b/+Efqwz2ZlwGcj4Dx.Igbx9vG0
   mftIIxM4TUX28KBbaLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhg1FU8.DQiQEki
   nn66GPMtm1SNgitxFxWouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhk.g0gMIQ
```

```
TXAgMBAAECggEANlrz8XNX/lxBELixK0H83o4aYKYqDKZfZkUN8hU33xpu.Y/0sc
VbLbu46WzysoIfJFyUC+zFJnbMCCOPjGbI/4NWkEqc9TAlKz+wDo+hf5bf0.ypFr
EmikHk8R3fkpnvKi69ldw0iYnqcFVhq7VtGrSmJcy6Hckwbk7EBoUZGL0wtp.xlO
6XlhksAvn8+75qoWzsNhi7S/L0IVCVLbUaV3hodTHlH5M4daFbqyRWD7UiPKt.Q3
hdw1rpyVZg8ZbBFp0Ej4f9GdRaq88SIKMKCDu3t9ibn/v1kEte+PxhuwyW+d0o.h
kKSEW0yLKCzQm5tujsPq0UVzPBkLJACUnFAi+a4AQKBgQDu6VLH2eYoTjPPTyAv.
vOJnNWP7oMzyJ4/eFqdE9m+2Ajm/0qaMY95ftZ+GpEKggvC6Z5DFevEmgH4Sg2+G
.gFd93diyRPScVbNE8SmpXxLPU2UoykVmICuQZzLDNE18B3buxAm2GJ219NEnZOe
c.jPMOV/IcG1aLzTqQssL3zo/0gQKBgQDB4Olpg3EBggtJ/+dlkLHUw8c7Pe3UyL
kS.VxVsyQwioYt8xMeCWuPvPNFcOjcW53KN/YSpCVjpttKGsPtLibMlKYKgasEqg
cvl.Vb5OFtA/jNAP3mdAgCzBn6IF1NhVQe2dclo5puZ0gO38HDWq7EtqSi9Q0JSM
g3YC.QNcOORptVwKBgQCHrCafaYWDhA11/+g2U9x6Yd56ifF43rCbnV+2EQCVaqQ
i49xC.w4AH+Bs0mdlgT5unL6MOEmgZxkRR/SP7TKzixHYHnpMOqLhaQV24Wk5TQH
ek92D7.wu8aXRB9vBj4g0CuDNO6/jWpm/KenXXN+Fka3ySVg4zdbVmBzJJdqYckg
QKBgFXS.zSBzGgwz1/F7AaDZK49m1wPnhyeBb0OqHwbX/LI71rZ1mWef+nSF9Juh
/Y77B5/J.UPdO9vgGgS00nRk0LIRP2s5OU5IQgQTVLvf8a1UmbVgI+KX511Yi5yM
ztEwRcjEX.VM9ejXeXN0I57pvqG/xCOK3Kl2eYLh4TO9/E8WjjAoGAA1mqUV4Hnf
4yvF1rydMp.fpvoWekiiRE33iEbYZNATYhsl7uxwn760pqVifkq2DSrZeYm4+lw9
jwWMtUoPzpg.CJYMoGl846nhiZrbbJ5b5twoLV6GRmkk/CfOxPXNzCtSoQA86HHq
7rRdhXSau/bY.EXc91tnhLjFzZxdBgrd+f4k=.-----END PRIVATE KEY-----.
--estServerExampleBoundary.Content-Type: application/pkcs7-mime.
.-----BEGIN PKCS7-----.MIIDPAYJKoZIhvcNAQcCoIIDLTCCAykCAQExADALB
gkqhkiG9w0BBwGgggMPMIID.CzCCAfOgAwIBAgIBBTANBgkqhkiG9w0BAQUFADAX
MRUwEwYDVQQDEwxlc3RFeGFt.cGxlQ0EwHhcNMTIwNzA0MTgzOTM2WhcNMTMwNzA
0MTgzOTM2WjAsMSowKAYDVQQD.EyFzZXJ2ZXJzaWRlIGtleSBnZW5lcmF0ZWQgcm
VzcG9uc2UwggEiMA0GCSqGSIb3.DQEBAQUAA4IBDwAwggEKAoIBAQC0781l7tri0
yiiMb9ZZYch8zeizXrjMPF/Rxoz.2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+
DsSMtzxn4l+9tIsVDkAe4FyzN0hL.d/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImD
V3blvaMUKDSJhVbQ+z/G1W1TRx3iW.i5CMHYb+1pJXPTJz/GuWr/b/+Efqwz2Zlw
Gcj4DxIgbx9vG0mftIIxM4TUX28KBb.aLgJbalsiuOx3C2bEyaSPerdzqgvXFHGG
Ahg1FU8DQiQEkinn66GPMtm1SNgitxF.xWouFqpsax5MWn/i52TfEaF2PNThOuzK
tilweJhkg0gMIQTXAgMBAAGjTTBLMAkG.A1UdEwQCMAAwHQYDVR0OBBYEFLylcQN
0D5xTfRdayv+0GDULR2+EMB8GA1UdIwQY.MBaAFIR/SsVuU7I5IC+5INpMScsubQ
/zMA0GCSqGSIb3DQEBBQUAA4IBAQButIeM.DB9PkwlGGe7zqvUWVD8y99zowwV6A
rAOXWX+JO0bihgMtZaUfvPCX/LhZVEKDAki.W5orjAEvIu10b6l38ZzX2oyJgkYy
Mmbb14lzTsRyjiqFw9j1PXxwgZvhwcaCF4b7.eDUUBQIeZg3AnkQrEwnHR5oVIN5
8qo0P7PSKC3Vl3H6DlQh3y7w87nN12923/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-----
   MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC0781l7tri0yii
   Mb9ZZYch8zeizXrjMPF/Rxoz2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSM
   tzxn4l+9tIsVDkAe4FyzN0hLd/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImDV3blv
   aMUKDSJhVbQ+z/G1W1TRx3iWi5CMHYb+1pJXPTJz/GuWr/b/+Efqwz2ZlwGcj4Dx
   Igbx9vG0mftIIxM4TUX28KBbaLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhg1FU8
   DQiQEkinn66GPMtm1SNgitxFxWouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhk
   g0gMIQTXAgMBAAECggEANlrz8XNX/lxBELixK0H83o4aYKYqDKZfZkUN8hU33xpu
   Y/0scVbLbu46WzysoIfJFyUC+zFJnbMCCOPjGbI/4NWkEqc9TAlKz+wDo+hf5bf0
   ypFrEmikHk8R3fkpnvKi69ldw0iYnqcFVhq7VtGrSmJcy6Hckwbk7EBoUZGL0wtp
   xlO6XlhksAvn8+75qoWzsNhi7S/L0IVCVLbUaV3hodTHlH5M4daFbqyRWD7UiPKt
   Q3hdw1rpyVZg8ZbBFp0Ej4f9GdRaq88SIKMKCDu3t9ibn/v1kEte+PxhuwyW+d0o
   hkKSEW0yLKCzQm5tujsPq0UVzPBkLJACUnFAi+a4AQKBgQDu6VLH2eYoTjPPTyAv
   vOJnNWP7oMzyJ4/eFqdE9m+2Ajm/0qaMY95ftZ+GpEKggvC6Z5DFevEmgH4Sg2+G
   gFd93diyRPScVbNE8SmpXxLPU2UoykVmICuQZzLDNE18B3buxAm2GJ219NEnZOec
   jPMOV/IcG1aLzTqQssL3zo/0gQKBgQDB4Olpg3EBggtJ/+dlkLHUw8c7Pe3UyLkS
   VxVsyQwioYt8xMeCWuPvPNFcOjcW53KN/YSpCVjpttKGsPtLibMlKYKgasEqgcvl
   Vb5OFtA/jNAP3mdAgCzBn6IF1NhVQe2dclo5puZ0gO38HDWq7EtqSi9Q0JSMg3YC
   QNcOORptVwKBgQCHrCafaYWDhA11/+g2U9x6Yd56ifF43rCbnV+2EQCVaqQi49xC
   w4AH+Bs0mdlgT5unL6MOEmgZxkRR/SP7TKzixHYHnpMOqLhaQV24Wk5TQHek92D7
   wu8aXRB9vBj4g0CuDNO6/jWpm/KenXXN+Fka3ySVg4zdbVmBzJJdqYckgQKBgFXS
   zSBzGgwz1/F7AaDZK49m1wPnhyeBb0OqHwbX/LI71rZ1mWef+nSF9Juh/Y77B5/J
   UPdO9vgGgS00nRk0LIRP2s5OU5IQgQTVLvf8a1UmbVgI+KX511Yi5yMztEwRcjEX
   VM9ejXeXN0I57pvqG/xCOK3Kl2eYLh4TO9/E8WjjAoGAA1mqUV4Hnf4yvF1rydMp
   fpvoWekiiRE33iEbYZNATYhsl7uxwn760pqVifkq2DSrZeYm4+lw9jwWMtUoPzpg
   CJYMoGl846nhiZrbbJ5b5twoLV6GRmkk/CfOxPXNzCtSoQA86HHq7rRdhXSau/bY
   EXc91tnhLjFzZxdBgrd+f4k=
   -----END PRIVATE KEY-----
   --estServerExampleBoundary
   Content-Type: application/pkcs7-mime

   -----BEGIN PKCS7-----
   MIIDPAYJKoZIhvcNAQcCoIIDLTCCAykCAQExADALBgkqhkiG9w0BBwGgggMPMIID
   CzCCAfOgAwIBAgIBBTANBgkqhkiG9w0BAQUFADAXMRUwEwYDVQQDEwxlc3RFeGFt
   cGxlQ0EwHhcNMTIwNzA0MTgzOTM2WhcNMTMwNzA0MTgzOTM2WjAsMSowKAYDVQQD
   EyFzZXJ2ZXJzaWRlIGtleSBnZW5lcmF0ZWQgcmVzcG9uc2UwggEiMA0GCSqGSIb3
   DQEBAQUAA4IBDwAwggEKAoIBAQC0781l7tri0yiiMb9ZZYch8zeizXrjMPF/Rxoz
   2C9IU2THCrhPGXGQMne/zivce0m8/BMkkUc+DsSMtzxn4l+9tIsVDkAe4FyzN0hL
   d/zawgj6kUoCi3mxZnb2rWaRYAmM5w41ImDV3blvaMUKDSJhVbQ+z/G1W1TRx3iW
   i5CMHYb+1pJXPTJz/GuWr/b/+Efqwz2ZlwGcj4DxIgbx9vG0mftIIxM4TUX28KBb
   aLgJbalsiuOx3C2bEyaSPerdzqgvXFHGGAhg1FU8DQiQEkinn66GPMtm1SNgitxF
   xWouFqpsax5MWn/i52TfEaF2PNThOuzKtilweJhkg0gMIQTXAgMBAAGjTTBLMAkG
```

```
A1UdEwQCMAAwHQYDVR0OBBYEFLylcQN0D5xTfRdayv+0GDULR2+EMB8GA1UdIwQY
MBaAFIR/SsVuU7I5IC+5INpMScsubQ/zMA0GCSqGSIb3DQEBBQUAA4IBAQButIeM
DB9PkwlGGe7zqvUWVD8y99zowwV6ArAOXWX+JO0bihgMtZaUfvPCX/LhZVEKDAki
W5orjAEvIu10b6l38ZzX2oyJgkYyMmbb14lzTsRyjiqFw9j1PXxwgZvhwcaCF4b7
eDUUBQIeZg3AnkQrEwnHR5oVIN58qo0P7PSKC3Vl3H6DlQh3y7w87nN12923/wk0
v/bS3lv7lDX3HdmbQD1r2KPtBsJGF4jMdstT7FTx32ZFKObycbK7WJ4LHytNJDci
4iXf+B0S3D6Zbf1cXj80/W+jCGvU0+4SV3cgEXFE5VQvXd8x40W4h0dTSkQCDPOS
nPj4Dl/PsLqX3lDboQAxAA==
-----END PKCS7-----
--estServerExampleBoundary--
This is the epilogue. It is also to be ignored.
```

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