Internet Engineering Task Force (IETF)
Request for Comments: 7030
Category: Standards Track
ISSN: 2070-1721

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

October 2013

Enrollment over Secure Transport

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) certificates. It also supports client-generated public/private key pairs as well as key pairs generated by the CA.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7030.

Table of Contents

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], 1.2 [RFC5246], or any future version) and Hypertext Transfer Protocol (HTTP) [RFC2616] to provide an authenticated and authorized channel for Simple Public Key Infrastructure (PKI) Requests and Responses [RFC5272].

Architecturally, the EST service is located between a Certification Authority (CA) and a client. It performs several functions traditionally allocated to the Registration Authority (RA) 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 it 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 media types are used in conjunction with TLS. HTTPS operates over TCP; this document does not specify EST over HTTP/Datagram Transport Layer Security/User Datagram Protocol (HTTP/DTLS/UDP). With a suitable specification for combining HTTP, DTLS, and UDP, there are no EST requirements that would prevent it from working over such a stack. Figure 1 shows how the layers build upon each other.

EST Layering:

Protocols:

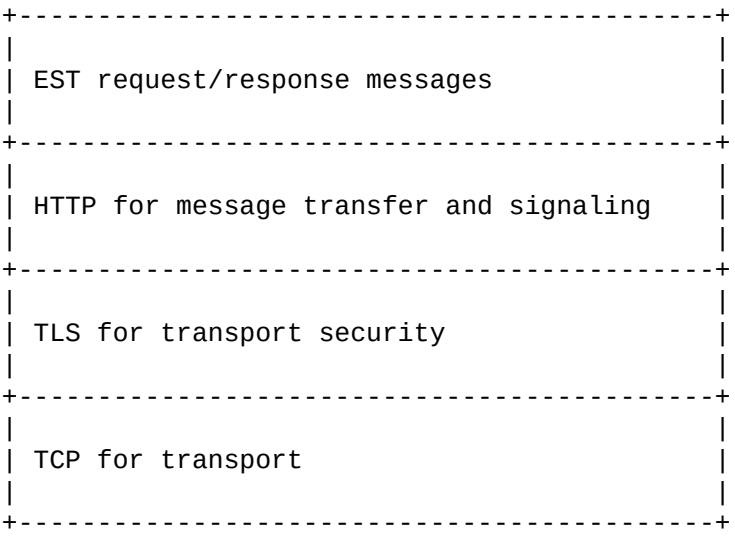

Figure 1

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "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 [RFC2986], 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: Any trust anchor (TA) that is not authoritative for the PKI hierarchy for which the EST server is providing services.

Explicit Trust Anchor: Any TA that is explicitly configured on the client or server 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 Sections 3.6 and 6.)

Implicit Trust Anchor: Any third-party TA that is available on the client or server 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 or TAs used by servers to authenticate manufacturer-installed client credentials (such as certificates populated into cable modems or routers in the factory). The authorization model for these TAs is different from the authorization model for Explicit Trust Anchors. (See more details in Sections 3.6.1, 3.6.2, and 6).

Certificate-Less TLS: Certificate-less TLS cipher suites provide a way to perform mutual authentication in situations where neither the client nor server have certificates or are willing to use them. 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 to an individual server.

2. Operational Scenario Overviews

This section provides an informative overview of the operational scenarios to better introduce the reader to the protocol discussion.

Both the EST clients and server are configured with information that provides the basis for mutual authentication and for authorization. The specific initialization data depends on the methods available in the client and server, but it 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 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 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 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 certificate(s) 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 different certificate than the one used to verify signatures on certificates and CRLs is used when EST protocol communication requires additional encryption.

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

- 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.

- 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.

- For bootstrapping, the EST client can rely upon manual authentication performed by the end-user as detailed in Section 4.1.1.

- The client can leverage the binding of a shared credential to a specific EST server with a certificate-less TLS cipher suite.

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 Sections 3.3.2, 3.3.3, and 3.7. The methods described in the normative text that are discussed in this overview include:

- TLS with a previously issued client certificate (e.g., an existing certificate issued by the EST CA);

- TLS with a previously installed certificate (e.g., manufacturer-installed certificate or a certificate issued by some other party);

- Certificate-less TLS (e.g., with a shared credential distributed out-of-band);

- 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. In all cases, HTTP-based client authentication is only to be performed over a TLS-protected transport (see Section 3.3).

2.3. Client Certificate Reissuance

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 reissued 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 attributes 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 Media Access Control (MAC) address of an interface of the EST client. 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:

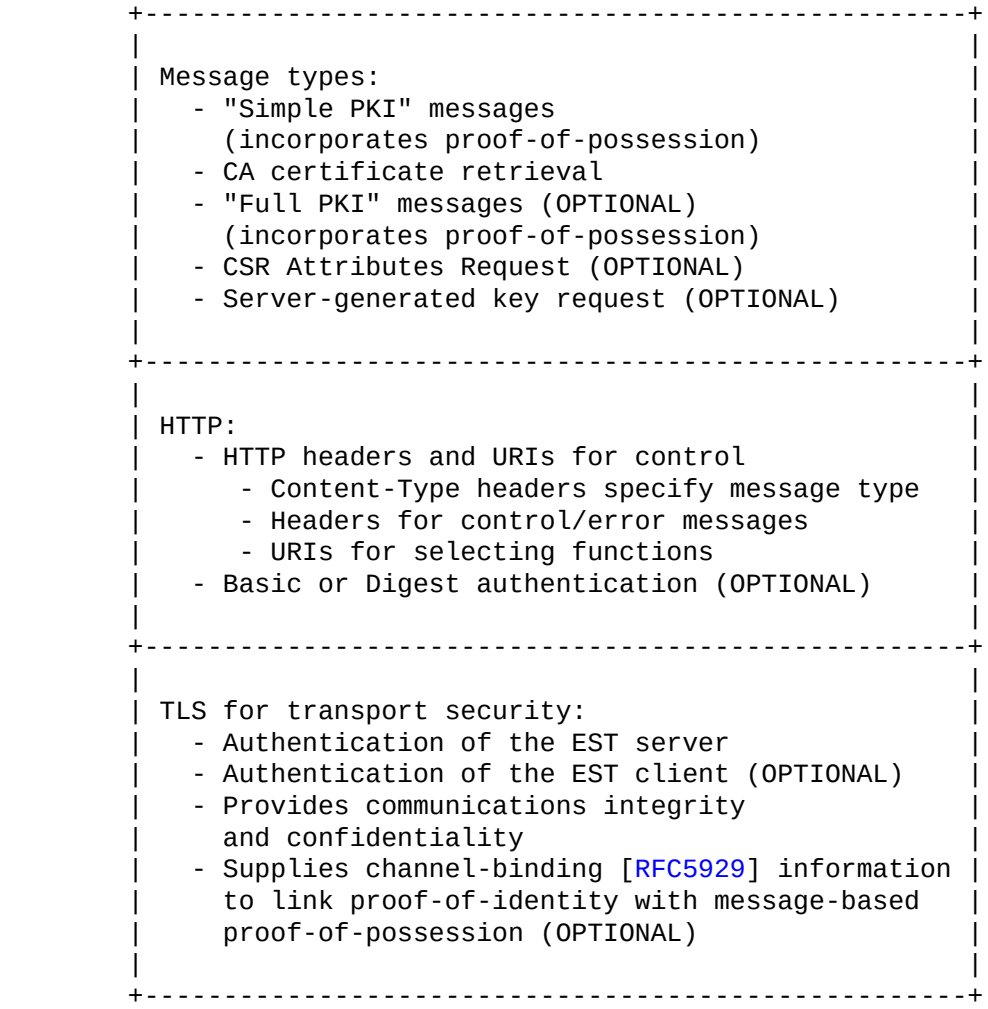

Figure 2

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 can 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 Initial Device Identifier (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 the CMC Transport Protocols [RFC5273]. CMC's POP and proof-of-identity mechanisms are defined in CMC, but the mechanisms here can also be used in conjunction with those mechanisms in "Full PKI" messages.

During protocol exchanges, different certificates can be used. The following table provides an informative overview. End-entities can 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. 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. Section 3.3.1 and Security Considerations
Third-party 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. Section 3.3.2
EST client certificate	The CA served by the EST server	Presented to the EST server during future EST operations. Section 3.3.2
End-entity certificate	The CA served by the EST server	Clients can obtain certs that are intended for non-EST uses. This includes certs that cannot be used for EST operations. Section 4.2.3

Figure 3

Trust anchor databases and their corresponding uses:

TA database	Use and section references
EST server Explicit 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. Section 3.3.2
EST server Implicit TA database	EST servers use this TA database to authenticate certificates issued by third-party TAs; e.g., EST client certificates issued by a device manufacturer. An Implicit TA database can be disabled. Section 3.3.2
EST client Explicit TA database	EST clients use this TA database to authenticate certificates issued by the EST CA, including EST server certificates. Sections 3.1, 3.3.1, 3.6.1, and 4.1.1
EST client Implicit TA database	EST clients use this TA database to authenticate an EST server that uses an externally issued certificate. An Implicit TA database can be disabled. Sections 3.1, 3.3.1, 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 6. 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" and "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 the lack of a client certificate suitable for use by TLS (see 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 RFC 2616, persistent connections may be used to reduce network and processing loads associated with multiple HTTP requests. EST does not require or preclude persistent HTTP connections.

3.2.1. HTTP Headers for Control

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 specified in the HTTP Content-Type header indicate which EST message is being transferred. Media types used by EST are specified in Section 3.2.4.

HTTP redirections (3xx status codes) to the same web origin (see [RFC6454]) SHOULD be handled by the client without user input so long as all applicable security checks (Sections 3.3 and 3.6) have been enforced on the initial connection. The client initiates a new TLS connection and performs all applicable security checks when redirected to other web origin servers. Redirections to other web origins require the EST client to obtain user input for non-GET or HEAD requests as specified in [RFC2616]. Additionally, if the client has already generated a CSR that includes linking identity and POP information (Section 3.5), then the CSR will need to be recreated to incorporate the tls-unique from the new, redirected session. Note: the key pair need not be regenerated. These are processing and interface burdens on the client. EST server administrators are advised to take this into consideration.

3.2.2. HTTP URIs for Control

The EST server MUST support the use of the path-prefix of "/.well-known/" as defined in [RFC5785] 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 Clients (MUST)	/simpleenroll	Section 4.2
Re-enrollment of Clients (MUST)	/simplereenroll	Section 4.2.2
Full CMC (OPTIONAL)	/fullcmc	Section 4.3
Server-Side Key Generation (OPTIONAL)	/serverkeygen	Section 4.4
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-line:

```
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-line:

```
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 regardless of whether the additional path segment is present. 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 [RFC5272] uses the same messages for certificate renewal and certificate rekey.

An EST server can 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. NULL and anon cipher suites MUST NOT be used because they do not provide confidentiality or support mutual certificate-based or certificate-less authentication, respectively. As specified in "Certificate Management over CMS (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 6. The client MUST NOT respond to the server's HTTP authentication request unless the client has authorized 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 FTP and HTTP [RFC2585], application/pkcs10 [RFC5967], and CMC [RFC5272].

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 for each operation are:

Message type (per operation)	Request media type Response media type(s) Source(s) of types	Request section(s) Response section
Distribution of CA Certificates /cacerts	N/A application/pkcs7-mime [RFC5751]	Section 4.1 Section 4.1.1
Client Certificate Request Functions /simpleenroll /simplereenroll	application/pkcs10 application/pkcs7-mime [RFC5967] [RFC5751]	Sections 4.2/4.2.1 Section 4.2.2
Full CMC /fullcmc	application/pkcs7-mime application/pkcs7-mime [RFC5751]	Section 4.3.1 Section 4.3.2
Server-Side Key Generation /serverkeygen	application/pkcs10 multipart/mixed (application/pkcs7-mime & application/pkcs8) [RFC5967] [RFC5751] [RFC5958]	Section 4.4.1 Section 4.4.2
CSR Attributes /csrattrs	N/A application/csrattrs (This document)	Section 4.5.1 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 mutual authentication has been performed. TLS authentication is most commonly enabled with the use of certificates [RFC5280]. Alternately, certificate-less TLS authentication, where neither the client nor server present a certificate, is also an acceptable method for EST mutual authentication (Section 3.3.3). 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).

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 for all EST communications. TLS session resumption [RFC5077] SHOULD be supported.

TLS channel-binding information can 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. The EST server MUST implement Section 3.5.

3.3.1. TLS-Based Server Authentication

TLS server authentication with certificates MUST be supported.

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).

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.

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 client certificate issued by a third party to authenticate itself.

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

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

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. Transport Layer Security-Secure Remote Password (TLS-SRP) cipher suites, i.e., those with _SRP_ in the name, listed in Section 2.7 of [RFC5054] are suitable for this purpose. Section 6 lists the characteristics of a cipher suite 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 that 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 of 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 clients are required to use the mechanism specified in this section. This specification provides a 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 SHOULD 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 "Linking Identity and POP Information" method defined by Section 6 of [RFC5272] that is available if Full PKI messages are used.

The client generating the CSR obtains the tls-unique value from the TLS subsystem as described in Channel Bindings for TLS [RFC5929]. The EST client operations between obtaining the tls-unique value through generation of the CSR that contains the current tls-unique value and the subsequent verification of this value by the EST server are the "phases of the application protocol during which application-layer authentication occurs"; these operations are protected by the synchronization interoperability mechanism described in the "Channel Bindings for TLS" interoperability notes in Section 3.1 of [RFC5929].

When performing renegotiation, TLS "secure_renegotiation" [RFC5746] MUST be used.

The tls-unique value is base64 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). The challenge-password field is limited to 255 bytes (Section 7.4.9 of [RFC5246] indicates that no existing cipher suite would result in an issue with this limitation). If the challenge-password attribute is absent, the client did not include the optional channel-binding information (the presence of the challenge-password attribute indicates the inclusion of 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 [RFC5272] "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 the 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 or the most recent HTTP redirection URI against the server's identity according to the rules specified in [RFC6125], Section 6.4, 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 configured URI and each HTTP redirection URI according to the rules specified in [RFC6125], Section 6.4. The provisioned URI or the most recent HTTP redirection 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 an Implicit TA database that allows a bootstrap installation of the Explicit TA database as described in 4.1.3. This section describes an alternate 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 an 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 (Sections 4.1.3 or 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 (CMC [RFC5272], Section 6.15) 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 as specified in Section 3.3.1 or if the optional certificate-less authentication is used as specified in Section 3.3.3.

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. CA Certificates Request

EST clients request the EST CA 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 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 if certificate-based authentication is used or Section 3.3.3 if the optional certificate-less authentication is used, 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. CA Certificates Response

If successful, the server response MUST have an HTTP 200 response code. Any other response code indicates an error and the client MUST abort the protocol.

A successful response MUST be a certs-only CMC Simple PKI Response, as defined in [RFC5272], containing the certificates described in the following paragraph. The HTTP content-type of "application/pkcs7-mime" is used. The Simple PKI Response is sent with a Content-Transfer-Encoding of "base64" [RFC2045].

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 reinitialized with the PKI using the Bootstrap Distribution of CA certificates (Section 4.1.1) method, which involves user interaction.

After out-of-band 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 disables the Implicit TA database, and if the EST server certificate was verified using an Implicit TA database entry, then the client MUST include the "Trusted CA Indication" extension in future TLS sessions [RFC6066]. This indicates to the server that only an EST server certificate authenticatable by the Explicit TA database entry is now acceptable (otherwise, the EST server might continue to use a server certificate that is only verifiable by a now disabled Implicit TA).

The EST client SHOULD also make the CA Certificate response information available to the end-entity software for use when validating peer certificates.

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 if certificate-based authentication is used or Section 3.3.3 if the optional certificate-less authentication is used. The client MUST verify the authorization of the EST server as specified in Section 3.6.

The server MUST authenticate the client as specified in Section 3.3.2 if certificate-based authentication is used or Section 3.3.3 if the optional certificate-less authentication is used. 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 [RFC5272], Section 3.1 (i.e., a PKCS #10 Certification Request [RFC2986]).

The Certification Signing Request (CSR) signature provides proof-of-possession of the client-possessed private key to the EST server. If the CSR KeyUsage extension indicates that the private key can be used to generate digital signatures, then the client MUST generate the CSR signature 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 MAY still be generated using the private key, but the key MUST NOT be used 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 are used when the key pair cannot 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 [RFC5967] with a Content-Transfer-Encoding of "base64" [RFC2045].

If the EST client authenticated using a previously installed certificate issued by a third-party CA (see Section 2.2.1), the client MAY include the ChangeSubjectName attribute, as defined in [RFC6402], in the CSR to request that the subjectName and SubjectAltName be changed in the new certificate.

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, then the EST server renews the client certificate. If the public key information in the certification request is different than the current client certificate, then the EST server rekeys the client certificate.

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".

A successful response MUST be a certs-only CMC Simple PKI Response, as defined in [RFC5272], containing only the certificate that was issued. The HTTP content-type of "application/pkcs7-mime" with an smime-type parameter "certs-only" is used, as specified in [RFC5273].

The server MUST answer with a suitable 4xx or 5xx HTTP [RFC2616] error code when a problem occurs. 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 plaintext human-readable error message containing explanatory 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 states 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].

If the client closes the TLS connections while waiting for the Retry-After time to expire, then the client initiates a new TLS connection and performs all applicable security checks. If the client has already generated a CSR that includes linking identity and POP information (Section 3.5), then the CSR will need to be recreated to incorporate the tls-unique from the new, redirected session. Note: the key pair need not be regenerated. These are processing and interface burdens on the client. EST server administrators are advised to take this into consideration.

The EST client MAY also make the certificate response, and associated private key, available to end-entity software for use as an end-entity certificate.

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" with an smime-type parameter "CMC-request", as specified in [RFC5273]. The body of the message is the binary value of the encoding of the PKI Request with a Content-Transfer-Encoding of "base64" [RFC2045].

4.3.2. Full CMC 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 with an smime-type parameter of "certs-only" or the Full PKI Response with an smime-type parameter "CMC-response", as specified in Section 3.2.1 of [RFC5751]. The body of the message is the binary value of the encoding of the PKI Response with a Content-Transfer-Encoding of "base64" [RFC2045].

When rejecting a request, the server MUST specify either an HTTP 4xx error or an HTTP 5xx error. A CMC response with the content-type of "application/pkcs7-mime" MUST be included in the response data for any CMC error response.

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.

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 if certificate-based authentication is used or Section 3.3.3 if the optional certificate-less authentication is used, and check the server's authorization as given in Section 3.6.

The EST server MUST authenticate the client, as specified in Section 3.3.2 if certificate-based authenticated is used or Section 3.3.3 if the optional certificate-less authentication is used, and check the client's authorization as given in Section 3.7. The EST server applies whatever authorization or logic it chooses to determine if the private key and certificate should be provided.

Cipher suites that have a NULL confidentiality algorithm MUST NOT be used as they will disclose the contents of an unprotected private key.

Proper random number and key generation [RFC4086] is a server implementation responsibility, and server archiving of generated keys is determined by CA policy. 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 "/simplereenroll" path extensions with the same content-type and transfer encoding.

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 of 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 an attribute in the CSR indicating the encryption key to be used. Both symmetric and asymmetric encryption are supported as described in the following subsections. The client MUST also include an SMIMECapabilities attribute ([RFC2633], Section 2.5) in the CSR to indicate the key encipherment algorithms the client is willing to use.

It is strongly RECOMMENDED that the clients request that the returned private key be afforded the additional security of the Cryptographic Message Syntax (CMS) EnvelopedData in addition to the TLS-provided security to protect against unauthorized disclosure.

4.4.1.1. Requests for Symmetric Key Encryption of the Private Key

To specify a symmetric encryption key to be used to encrypt the server-generated private key, the client MUST include a DecryptKeyIdentifier attribute (as defined in Section 2.2.5 of [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 secret key to encrypt a private 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.

4.4.1.2. Requests for Asymmetric Encryption of the Private Key

To specify an asymmetric encryption key to be used to encrypt the server-generated private key, the client MUST include an AsymmetricDecryptKeyIdentifier attribute. The AsymmetricDecryptKeyIdentifier attribute is defined as:

```
id-aa-asymmDecryptKeyID OBJECT IDENTIFIER ::= {
    id-aa 54 }
```

The asymmetric-decrypt-key-identifier attribute values have ASN.1 type AsymmetricDecryptKeyIdentifier (where ASN.1 is defined in [X.680])::

```
   AsymmetricDecryptKeyIdentifier ::= OCTET STRING
```

If the server does not have a public 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 AsymmetricDecryptKeyIdentifier to the key generator and the client is outside the scope of this document. If the key identified is bound to an X.509 certificate, then the key MUST either explicitly support keyTransport or keyAgreement or its use MUST be unrestricted.

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.

If additional encryption is not being employed, the private key data MUST be placed in an "application/pkcs8". An "application/pkcs8" part consists of the base64-encoded DER-encoded [X.690] PrivateKeyInfo with a Content-Transfer-Encoding of "base64" [RFC2045].

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 or the EST CA. The SignedData is further protected by placing it inside of a CMS EnvelopedData, as described in Section 4 of [RFC5958]. The following list shows how the EncryptedData is used, depending on the type of protection key specified by the client.

- If the client specified a symmetric encryption key to protect the server-generated private key, the EnvelopedData content is encrypted using the secret key identified in the request. The EnvelopedData RecipientInfo field MUST indicate the key-encryption kekri key management technique. The values are as follows: version is set to 4, key-encryption key identifier (kekid) is set to the value of the DecryptKeyIdentifier from Section 4.4.1.1; keyEncryptionAlgorithm is set to one of the key wrap algorithms that the client included in the SMIMECapabilities accompanying the request; and encryptedKey is the encrypted key.

- If the client specified an asymmetric encryption key suitable for key transport operations to protect the server-generated private key, the EnvelopedData content is encrypted using a randomly generated symmetric encryption key. The cryptographic strength of the symmetric encryption key SHOULD be equivalent to the client-specified asymmetric key. The EnvelopedData RecipientInfo field MUST indicate the KeyTransRecipientInfo (ktri) key management technique. In KeyTransRecipientInfo, the RecipientIdentifier (rid) is either the subjectKeyIdentifier copied from the attribute defined in Section 4.4.1.2 or the server determines an associated issuerAndSerialNumber from the attribute; version is derived from the choice of rid [RFC5652], keyEncryptionAlgorithm is set to one of the key wrap algorithms that the client included in the SMIMECapabilities accompanying the request, and encryptedKey is the encrypted key.

- If the client specified an asymmetric encryption key suitable for key agreement operations to protect the server-generated private key, the EnvelopedData content is encrypted using a randomly generated symmetric encryption key. The cryptographic strength of the symmetric encryption key SHOULD be equivalent to the client-specified asymmetric key. The EnvelopedData RecipientInfo field MUST indicate the KeyAgreeRecipientInfo (kari) key management technique. In the KeyAgreeRecipientInfo type, version, originator, and user keying material (ukm) are as in [RFC5652], and keyEncryptionAlgorithm is set to one of the key wrap algorithms that the client included in the SMIMECapabilities accompanying the request. The recipient's key identifier is either copied from the attribute defined in Section 4.4.1.2 to subjectKeyIdentifier or the server determines an IssuerAndSerialNumber that corresponds to the value provided in the attribute.

In all three additional encryption cases, the EnvelopedData is returned in the response as an "application/pkcs7-mime" part with an smime-type parameter of "server-generated-key" and a Content-Transfer-Encoding of "base64".

The certificate data part is an "application/pkcs7-mime" and exactly matches the certificate response to /simpleenroll.

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 plaintext 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.

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

Requesting CSR attributes is optional, but clients are advised that CAs 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.

Responses to attribute request messages MUST be encoded as the content-type of "application/csrattrs" with a Content-Transfer-Encoding of "base64" [RFC2045]. The syntax for application/csrattrs body is as follows:

```
CsrAttrs ::= SEQUENCE SIZE (0..MAX) OF AttrOrOID

AttrOrOID ::= CHOICE (oid OBJECT IDENTIFIER, attribute Attribute }

Attribute { ATTRIBUTE:IOSet } ::= SEQUENCE {
     type   ATTRIBUTE.&id({IOSet}),
     values SET SIZE(1..MAX) OF ATTRIBUTE.&Type({IOSet}{@type}) }
```

An EST server includes zero or more OIDs or attributes [RFC2986] that it requests the client to use in the certification request. The client MUST ignore any OID or attribute it does not recognize. When the server encodes CSR Attributes as an empty SEQUENCE, it means that the server has no specific additional information it desires in a client certification request (this is functionally equivalent to an HTTP response code of 204 or 404).

If the CA requires a particular crypto system or use of a particular signature scheme (e.g., certification of a public key based on a certain elliptic curve, or signing using a certain hash algorithm) it MUST provide that information in the CSR Attribute 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.

The structure of the CSR Attributes Response SHOULD, to the greatest extent possible, reflect the structure of the CSR it is requesting. Requests to use a particular signature scheme (e.g. using a particular hash function) are represented as an OID to be reflected in the SignatureAlgorithm of the CSR. Requests to use a particular crypto system (e.g., certification of a public key based on a certain elliptic curve) are represented as an attribute, to be reflected as the AlgorithmIdentifier of the SubjectPublicKeyInfo, with a type indicating the algorithm and the values indicating the particular parameters specific to the algorithm. Requests for descriptive information from the client are made by an attribute, to be represented as Attributes of the CSR, with a type indicating the [RFC2985] extensionRequest and the values indicating the particular attributes desired to be included in the resulting certificate's extensions.

The sequence is Distinguished Encoding Rules (DER) encoded [X.690] and then base64 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 challengePassword (indicating that linking of identity and POP information is requested; see Section 3.5), an extensionRequest with the Media Access Control (MAC) address ([RFC2307]) of the client, and to use the secp384r1 elliptic curve and to sign with the SHA384 hash function. Then, it takes the following:

```
OID:         challengePassword (1.2.840.113549.1.9.7)

Attribute:   type = extensionRequest (1.2.840.113549.1.9.14)
             value = macAddress (1.3.6.1.1.1.1.22)

Attribute:   type = id-ecPublicKey (1.2.840.10045.2.1)
             value = secp384r1 (1.3.132.0.34)

OID:         ecdsaWithSHA384 (1.2.840.10045.4.3.3)
```

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

```
30 41 06 09 2a 86 48 86 f7 0d 01 09 07 30 12 06 07 2a 86 48 ce 3d
02 01 31 07 06 05 2b 81 04 00 22 30 16 06 09 2a 86 48 86 f7 0d 01
09 0e 31 09 06 07 2b 06 01 01 01 01 16 06 08 2a 86 48 ce 3d 04 03
03
```

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

```
MEEGCSqGSIb3DQEJBzASBgcqhkjOPQIBMQcGBSuBBAAiMBYGCSqGSIb3DQEJDjEJ
BgcrBgEBAQEWBggqhkjOPQQDAw==
```

5. IANA Considerations

Section 4.4.1.2 defines an OID that has been registered in an arc delegated by the IANA to the PKIX working group.

IANA has registered the following:

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

```
URI suffix: est

Change controller: IETF
```

IANA has updated the "Application Media Types" registry with the following filled-in templates from [RFC6838].

The media subtype for CSR attributes in a CSR Attributes Response 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 is normally 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: .csrattrs

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 <iesg@ietf.org>

The application/pkcs7-mime content-type defines the optional "smime-type" parameter [RFC5751] with a set of specific values. This document adds another value, "server-generated-key", as the parameter value for Server-side Key Generation Response.

6. 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 responder that is certified by a third party for the URI configured, but it 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 use only 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 man in the middle). 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 cipher suites that maintain security and perform the mutual authentication necessary for enrollment have the following properties:

- the only information leaked by an active attack is whether or not a single guess of the secret is correct.
- any advantage an adversary gains is through interaction and not computation.
- 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 cipher suite 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.

TLS cipher suites that include "_EXPORT_" and "_DES_" in their names MUST NOT be used. These ciphers do not offer a sufficient level of protection; 40-bit crypto in 2013 doesn't offer acceptable protection, and the use of DES is deprecated.

As described in CMC, Section 6.7 of [RFC5272], "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 without any application-layer protection. 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 RECOMMENDED 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.

ASN.1 encoding rules (e.g., DER and BER) have a type-length-value structure, and it is easy to construct malicious content with invalid length fields that can cause buffer overrun conditions. ASN.1 encoding rules allow for arbitrary levels of nesting, which may make it possible to construct malicious content that will cause a stack overflow. Interpreters of ASN.1 structures should be aware of these issues and should take appropriate measures to guard against buffer overflows and stack overruns in particular, and malicious content in general.

7. References

7.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, 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.

[RFC2633] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[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.

[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.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

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

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 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.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[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.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[X.680] ITU-T Recommendation X.680 (2008) | ISO/IEC 8824-1:2008, "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 X.690 (2008) | ISO/IEC 8825-1:2008, "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 Standards Association, "IEEE 802.1AR Secure Device Identifier", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[RFC2307] Howard, L., "An Approach for Using LDAP as a Network Information Service", RFC 2307, March 1998.

[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.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, November 2007.

[RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, August 2010.

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

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", Federal Information Processing Standard Publication 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[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>.

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 /.well-known/est/cacerts 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: 192.0.2.1:8085
Accept: */*
```

In response, the server provides the current CA certificates:

```
HTTP/1.1 200 OK
Status: 200 OK
Content-Type: application/pkcs7-mime
Content-Transfer-Encoding: base64
Content-Length: 4246

MIIMOQYJKoZIhvcNAQcCoIIMKjCCDCYCAQExADALBgkqhkiG9w0BBwGgggwMMIIC
+zCCAeOgAwIBAgIJAJpY3nUZO3qcMA0GCSqGSIb3DQEBBQUAMBsxGTAXBgNVBAMT
EGVzdEV4YW1wbGVDQSBPd08wHhcNMTMwNTA5MDM1MzMxWhcNMTQwNTA5MDM1MzMx
WjAbMRkwFwYDVQQDExBlc3RFeGFtcGxlQ0EgT3dPMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAwDqpiHopaICubpRqbpEN7LqTIqWELFIA9qDDheHIKuyO
HW/ZAP7Rl4S5ZU6gaLW/ksseBUxdmox3KNyvtyjehIofTu28eZWhgy6/LCEGWR3P
K+fgPBA0l0JfJR/8oeXZa70oLVQc3hI4kCeqjFMs+biYH0vp/RluhftyZ5kzQyH1
EGsRkw1/qUKkTZ8PCF8VFlYfqmUoqsaRTyZbjII4J+Y6/jEG+p7QreW9zcz4sPe8
3c/uhwMLOWQkZtKsQtgo5CpfYMjuAmk4Q2joQq2vcxlc+WNKHf+wbrDb11ORZril
9ISlI94oumcRz3uBG1Yg7z83hdDfasmdfbp8gOSNFQIDAQABo0IwQDAPBgNVHRMB
Af8EBTADAQH/MB0GA1UdDgQWBBQITTKxMqATXrfc4ffpCIbt6Gsz0jAOBgNVHQ8B
Af8EBAMCAQYwDQYJKoZIhvcNAQEFBQADggEBACPnQPu5WReUGuCMS0nBOGa2tXh6
uZP4mS3J1qEfDePam/IiU9ssyYdcDwhVvKMoP4gI/yu4XFqhdpIoy/PyD4T15MT7
KADCxXkh5rM1IqMui7FvBKLWYGdy9sjEf90wAkBjHBe/TMO1NNw3uELyONSkHIvo
X0pu6aPmm/moIMyGi46niFse1iWlXXldGLkOQsh0e7U+wpBX07QpOr2KB2+Yf+uA
KY1SWzEG23bUxXlvcbUMgANDGj5r6z+niKL0VlApip/iCuVEEOcZ91UlmJjVLQWA
x6ie+v84oM+pIojiGM0C4XWcVlKKEgcMOsN3S4lvm8Ptpq0GLoIJY8NTD20wggMD
MIIB66ADAgECAgEBMA0GCSqGSIb3DQEBBQUAMBsxGTAXBgNVBAMTEGVzdEV4YW1w
bGVDQSBPd08wHhcNMTMwNTA5MDM1MzMyWhcNMTQwNTA5MDM1MzMyWjAbMRkwFwYD
```

```
VQQDExBlc3RFeGFtcGxlQ0EgTndPMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAnn3rZ3rMJHwf7MD9K4mubxHAvtdnrsQf5OfgtMhRIL4aePNhAdgPyj8C
loxOgD3UTV+dQ1ViOzVxPN7acikoOnkIdRpjpOpkyMo+KkvHMQXGnQTbsMAv1qWt
9S12DMpo0GOA1e4Ge3ud5YPOTR/q6PvjN51IEwYKksG7CglwZwB+5JbwhYr2D/0u
btGltriRVixPWrvt+wz/ITp5rcjh/8RS3LE8tQy3kTNhJF3Y/esR2sSgOiPNgIto
CATysbaINEPr4MemqML4tDpR/aG9y+8Qe7s1LyMFvDletp2mmBykAC/7nOat/pwU
lB0sN524D1XAgz8ZKvWrkh+ZaOr3hwIDAQABo1IwUDAOBgNVHQ8BAf8EBAMCBLAw
HQYDVR0OBBYEFLHEaeZbowSn2Jejizu/uWqyMkI8MB8GA1UdIwQYMBaAFAhNMrEy
oBNet9zh9+kIhu3oazPSMA0GCSqGSIb3DQEBBQUAA4IBAQCLDkL7aLNV6hSOkIqH
q+shV9YLO56/tj00vY/jV5skgDHk5d0B+OGortKVuGa57+v0avTrlJns3bNW8Ntv
zkDEhmd00Ak02aPsi4wRHLFgttUf9HdEHAuTkAESPTU43DiptjkfHhtBMfsFrCkd
sxWzCz+prDOMHYfUEkhRVV++1zyGEX6ov1Ap2IU2p3E+ASihL/amxTEQAsbwjUTI
R52zoL6nMPzpbKeZi2M0eEBVF8sDueA9Hjo6woLjgJqV0/yc5vC2HAxUOhx0cWTY
GcRBgL/yOyQLKiY5TKBH951OjQ4vhF2HmcoO7DkcNLYJOge16ssx4ogBHul20VgF
XJJjMIIDAzCCAeugAwIBAgIBAjANBgkqhkiG9w0BAQUFADAbMRkwFwYDVQQDExBl
c3RFeGFtcGxlQ0EgTndOMB4XDTEzMDUwOTAzNTMzMloXDTE0MDUwOTAzNTMzMlow
GzEZMBcGA1UEAxMQZXN0RXhhbXBsZUNBIE93TjCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAMA6qYh6KWiArm6Uam6RDey6kyKlhCxSAPagw4XhyCrsjh1v
2QD+0ZeEuWVOoGi1v5LLHgVMXZqMdyjcr7co3oSKH07tvHmVoYMuvywhBlkdzyvn
4DwQNJdCXyUf/KHl2Wu9KC1UHN4SOJAnqoxTLPm4mB9L6f0ZboX7cmeZM0Mh9RBr
EZMNf6lCpE2fDwhfFRZWH6plKKrGkU8mW4yCOCfmOv4xBvqe0K3lvc3M+LD3vN3P
7ocDCzlkJGbSrELYKOQqX2DI7gJpOENo6EKtr3MZXPljSh3/sG6w29dTkWa4pfSE
pSPeKLpnEc97gRtWIO8/N4XQ32rJnX26fIDkjRUCAwEAAaNSMFAwDgYDVR0PAQH/
BAQDAgSwMB0GA1UdDgQWBBQITTKxMqATXrfc4ffpCIbt6Gsz0jAfBgNVHSMEGDAW
gBSxxGnmW6MEp9iXo4s7v7lqsjJCPDANBgkqhkiG9w0BAQUFAAOCAQEALhDaE6Mp
BINBsJozdbXlijrWxL1CSv8f4GwpUFk3CgZjibt/qW9UoaNR4E58yRopuEhjwFZK
2w8YtRqx8IZoFhcoLkpBDfgLLwhoztzbYvOVKQMidjBlkBEVNR5MWdrs7F/AxWuy
iZ2+8AnR8GwqEIbCD0A7xIghmWEMh/BVI9C7GLqd6PxKrTAjuDfEpfdWhU/uYKmK
cL3XDbSwr30j2EQyaTV/3W0Tn2UfuxdwDQ4ZJs9G+Mw50s7AG6CpISyOIFmX6/bU
DpJXGLiLwfJ9C/aum9nylYuGCJ68BuTrCs9567KGfXEXI0mdFFCL7TaVR43kjsg3
c43kZ7369MeEZzCCAvswggHjoAMCAQICCQDprp3DmjOyETANBgkqhkiG9w0BAQUF
ADAbMRkwFwYDVQQDExBlc3RFeGFtcGxlQ0EgTndOMB4XDTEzMDUwOTAzNTMzMloX
DTE0MDUwOTAzNTMzMlowGzEZMBcGA1UEAxMQZXN0RXhhbXBsZUNBIE53TjCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJ5962d6zCR8H+zA/SuJrm8RwL7X
Z67EH+Tn4LTIUSC+GnjzYQHYD8o/ApaMToA91E1fnUNVYjs1cTze2nIpKDp5CHUa
Y6TqZMjKPipLxzEFxp0E27DAL9alrfUtdgzKaNBjgNXuBnt7neWDzk0f6uj74zed
SBMGCpLBuwoJcGcAfuSW8IWK9g/9Lm7Rpba4kVYsT1q77fsM/yE6ea3I4f/EUtyx
PLUMt5EzYSRd2P3rEdrEoDojzYCLaAgE8rG2iDRD6+DHpqjC+LQ6Uf2hvcvvEHu7
NS8jBbw5XradppgcpAAv+5zmrf6cFJQdLDeduA9VwIM/GSr1q5IfmWjq94cCAwEA
AaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUscRp5lujBKfYl6OLO7+5
arIyQjwwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBBQUAA4IBAQBCz/CWdYvn
GM/SdCdEiom5A1VxaW8nKgCWg/EyWtAIiaHQuViB+jTUAE9lona2MbJoFHW8U5e8
9dCP0rJpA9UYXXhWvFQzd5ZWpms4wUYt1j3gqqd36KorJIAuPigVng13yKytxM7c
VmxQnh0aux3aEnEyRGAhGalHp0RaKdgPRzUaGtipJTNBkSV5S4kD4yDCPHMNbBu+
OcluerwEpbz6GvE7CpXl2jrTBZSqBsFelq0iz4kk9++9CnwZwrVgdzklhRfJ1Z4j
NkLruwbQ+o4NvBZsXiKxNfn3K2o3SK8AuaEyDWkq18+5rjcfprRO8x4YTW+6mXPq
jM0MAGNDEW+1oQAxAA==
```

A.2. 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 for which the EST server provides services.

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

```
GET /.well-known/est/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: 192.0.2.1:8085
Accept: */*
```

In response, the server provides suggested attributes that are appropriate for the authenticated client. In this example, the EST server also includes two example attributes that the client would ignore unless the attribute type is known to the client:

```
HTTP/1.1 200 OK
Status: 200 OK
Content-Type: application/csrattrs
Content-Transfer-Encoding: base64
Content-Length: 171

MHwGBysGAQEBARYwIgYDiDcBMRsTGVBhcnNlIFNFVCBhcyAyLjk5OS4xIGRhdGEG
CSqGSIb3DQEJBzAsBgOINwIxJQYDiDcDBgOINwQTGVBhcnNlIFNFVCBhcyAyLjk5
OS4yIGRhdGEGCSskAwMCCAEBCwYJYIZIAWUDBAIC
```

A.3. Enroll/Re-enroll

The following is an example of a valid /simpleenroll exchange. The data messages for /simplereenroll are similar.

During this exchange, the EST client uses an out-of-band distributed username/password to authenticate itself to the EST server. This is the normal HTTP WWW-Authenticate behavior and is included here for informative purposes. When an existing TLS client certificate is used, the server might skip requesting the HTTP WWW-Authenticate header, such as during a /simplereenroll operation.

During the initial TLS handshake, the client can ignore the optional server-generated "certificate request" and can instead proceed with the HTTP POST request. In response to the initial HTTP POST attempt,

the server requests WWW-Authenticate from the client (this might occur even if the client used a client certificate, as detailed in the normative text above):

```
HTTP/1.1 401 Unauthorized
Content-Length: 0
WWW-Authenticate: Digest qop="auth", realm="estrealm",
nonce="1368141352"
```

In the subsequent HTTP POST, the username/password is included, along with the complete application/pkcs10 content:

```
POST /.well-known/est/simpleenroll HTTP/1.1
Authorization: Digest username="estuser", realm="estrealm", nonc
e="1368141352", uri="/.well-known/est/simpleenroll", cnonce="M
TYwMzg3", nc=00000001, qop="auth", response="144cc27f96046f1d70e
b16db20f75f22"
Host: 192.0.2.1:8085
Accept: */*
Content-Type: application/pkcs10
Content-Transfer-Encoding: base64
Content-Length: 882

MIIChTCCAW0CAQAwHzEdMBsGA1UEAxMUZGVtb3N0ZXA0IDEzNjgxNDEzNTIwggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQClNp+kdz+Nj8XpEp9kaumWxDZ3
eFYJpQKz9ddD5e5OzUeCm103ZIXQIxc0eVtMCatnRr3dnZRCAxGjwbqoB3eKt29/
XSQffVv+odbyw0WdkQOIbntCQry8YdcBZ+8LjI/N7M2krmjmoSLmLwU2V4aNKf0Y
MLR5Krmah3Ik31jmYCSvwTnv6mx6pr2pTJ82JavhTEIIt/fAYq1RYhkM1CXoBL+y
hEoDanN7TzC94skfS3VV+f53J9SkUxTYcy1Rw0k3VXfxWwy+cSKEPREl7I6k0YeK
tDEVAgBIEYM/L1S69RXTLujirwnqSRjOquzkAkD31BE961KZCxeYGrhxaR4PAgMB
AAGgITAfBgkqhkiG9w0BCQcxEhMQK3JyQ2lyLzcrRVl1NTBUNDANBgkqhkiG9w0B
AQUFAAOCAQEARBv0AJeXaHpl1MFIdzWqoi1dOCf6U+qaYWcBzpLADvJrPK1qx5pq
wXM830A1O+7RvrFv+nyd6VF2rl/MrNp+IsKuA9LYWIBjVe/LXoBO8dB/KxrYl16c
VUS+Yydi1m/a+DaftYSRGolMLtWeiqbc2SDBr2kHXW1TR130hIcpwmr29kC2Kzur
5thsuj276FGL1vPu0dRfGQfx4WWa9uAHBgz6tW37CepZsrUKe/0pfVhr2oHxApYh
cHGBQDQHVTFVjHccdUjAXicrtbsVhU5o1lPv7f4lEApv3SBQmJcaq5O832BzHw7n
PyMFcM15E9gtUVee5C62bVwuk/tbnGsbwQ==
```

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

```
HTTP/1.1 200 OK
Status: 200 OK
Content-Type: application/pkcs7-mime; smime-type=certs-only
Content-Transfer-Encoding: base64
Content-Length: 1122

MIIDOAYJKoZIhvcNAQcCoIIDKTCCAyUCAQExADALBgkqhkiG9w0BBwGgggMLMIID
BzCCAe+gAwIBAgIBFTANBgkqhkiG9w0BAQUFADAbMRkwFwYDVQQDExBlc3RFeGFt
cGxlQ0EgTndOMB4XDTEzMDUwOTIzMTU1M1oXDTE0MDUwOTIzMTU1M1owHzEdMBsG
A1UEAxMUZGVtb3N0ZXA0IDEzNjgxNDEzNTIwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQClNp+kdz+Nj8XpEp9kaumWxDZ3eFYJpQKz9ddD5e5OzUeCm103
ZIXQIxc0eVtMCatnRr3dnZRCAxGjwbqoB3eKt29/XSQffVv+odbyw0WdkQOIbntC
Qry8YdcBZ+8LjI/N7M2krmjmoSLmLwU2V4aNKf0YMLR5Krmah3Ik31jmYCSvwTnv
6mx6pr2pTJ82JavhTEIIt/fAYq1RYhkM1CXoBL+yhEoDanN7TzC94skfS3VV+f53
J9SkUxTYcy1Rw0k3VXfxWwy+cSKEPREl7I6k0YeKtDEVAgBIEYM/L1S69RXTLuji
rwnqSRjOquzkAkD31BE961KZCxeYGrhxaR4PAgMBAAGjUjBQMA4GA1UdDwEB/wQE
AwIEsDAdBgNVHQ4EFgQU/qDdB6ii6icQ8wGMXvy1jfE4xtUwHwYDVR0jBBgwFoAU
scRp5lujBKfYl6OLO7+5arIyQjwwDQYJKoZIhvcNAQEFBQADggEBACmxg1hvL6+7
a+lFTARoxainBx5gxdZ9omSb0L+qL+4PDvg/+KHzKsDnMCrcU6M4YP5n0EDKmGa6
4lY8fbET4tt7juJg6ixb95/760Th0vuctwkGr6+D6ETTfqyHnrbhX3lAhnB+0Ja7
o1gv4CWxh1I8aRaTXdpOHORvN0SMXdcrlCys2vrtOl+LjR2a3kajJO6eQ5leOdzF
QlZfOPhaLWen0e2BLNJI0vsC2Fa+2LMCnfC38XfGALa5A8e7fNHXWZBjXZLBCza3
rEs9Mlh2CjA/ocSC/WxmMvd+Eqnt/FpggRy+F8IZSRvBaRUCtGE1lgDmu6AFUxce
R4POrT2xz8ChADEA
```

A.4. 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. An example HTTP POSTed message is:

```
POST /.well-known/est/serverkeygen HTTP/1.1
Host: 192.0.2.1:8085
Accept: */*
Expect: 100-continue
Content-Type: application/pkcs10
Content-Transfer-Encoding: base64
Content-Length: 963

MIICwTCCAakCAQAwWzE+MDwGA1UEAxM1c2VydmVyS2V5R2VuIHJlcSBieSBjbGll
bnQgaW4gZGVtbyBzdGVwIDEyIDEzNjgxNDE5NTUxGTAXBgNVBAUTEFBJRDpXaWRn
ZXQgU046MTAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCvE1/6m4A/
3/L32Suyzbf28LM9y8CQfp0aepa7o20BSfluvm8HXR44mlV+wpieM8H5n3Ub3RIo
RUun/FllIzK9uV7UrkqJ3Yzmq2NOoTd4C+OPsV/RPTu873dhFrssDk3P4NIphlSS
sSIkt5rhz7wYbCqCFR5Aphe/30Jx7D+xBI5Rs8e6vRS8IpuImh71BHiLfhq9AFhz
4ZJsOUSVpUmqUogFsM7SOQ6XI4dl+djhpjT+YTJ6hQ2PXrqdch3KsTQ8c6aKs+e2
5QJxh7O8JHVlPHo4YIxXtAYSutcbbTN5TXWFCWSrWDJ+zuMmk2yU+dio1oW7YR7V
ftAvazJ3laQbAgMBAAGgITAfBgkqhkiG9w0BCQcxEhMQZEZzQVhtSm5qb2tCdER2
cjANBgkqhkiG9w0BAQUFAAOCAQEAR+I0EQB+hSjrLCAjnVH6bZdHUNGszIdwx1iu
L4n+0XK3SfEzeOMkC4T74yFGKj3redS1Ht9atYUPb0D1Qi9Jf9Co8eLblo1l19A6
GaS798ofxIF0Pl0Dr6/GqjheqJEIbcDTAJq+kvDihyQ4GQnhosygIZHvKppQlebA
gvp2RJSnMroPCe6RgTU9E2fmI9rin/9PyXeWFF1namp+lYbTGwjv1aE1ikhjCLlH
veHhCdgOExPw+fqhKhHjp+0ZKBlo2bC3pqRWvDTiZuwt9UpFFfGtuxvpTp44oS/j
M/965hWIw/5dshY/wQjIfYs07bbq2ERbpJiw9bAQY34gyoVmEQ==
```

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

```
HTTP/1.1 200 OK
Status: 200 OK
Content-Type: multipart/mixed ; boundary=estServerExampleBoundary
Content-Length: 3219

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
Content-Transfer-Encoding: base64

MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDPwKtwJ7TjMgA+
Poj64V909ryql0foP1hU4Yq5y8/bOP5ZTe6ArgVhUye099Ac+dfdwpyP/DESiujU
F/dS62Vck3UWNbnw+4O38FUp0enLbbjSTud48KpEW6+FzkeuAanPGZMA1wKyrYy9
rD5tQOOJU/CBVhUrITyYLZNYUe4agbpcR0wMtrRr2E58Mu8wQ80ryk6nkL7COk5Z
IQdNRxldk7DFvpA85Yn1stumoGRtVLW51iXeTS1LtXwhuUb/j6Lds3vvAiJ2SiZ0
Y3rxPlnJVyFmR8Mf2TBOjzuFqva/VLD2ayQjgaGEjq2ZWHXelQAOZ6N3lrChojEK
FGq93yOhAgMBAAECggEBALQ5az/nYjd5x9Y3f7NMUffwl+jRRfMHCMTRx/u4AEAo
KBYm0hFVZZtxfM+z7xlD8G0Th6gs2hFA6gwcIlUPmiX+UaOLxht0xWaLGgYmcNAm
BiCDjLBQ7xRQCWtlcK9WCA5+HBWtcEy6244rXxh+IyWd6NT6bXC165AEcX87y/e3
JFJ7XFNeDP656s2DmxSCci+iDte6SaEm7sJvYGu16qevJeMThcQcC9/rJjXkvpGL
IKK2px5idad4Pb6+QHpqj3d4oM8djO6wYUvrH8XQLqAaF8Hd5lFWVU57nSYY+H79
GaNDTfRTUL6AXr7kmMsKVFOJ0JjZExUCVMZtGiqhB6UCgYEA639OtdWLZCzyZFMe
p7VlRddoz0VAtrU2dxnEb4cWD8Gerg8uNvp8OG84gH+6MbPwz4ZYWKCgDFqyrAlw
SF02n9Sovh93eoJ5latSbfeYUkLtB8L/HVk5/CBGEsV9MUkdMF0+B43YlhyEDyKW
fX2+0UeHLFgRrfpSzP2cXduEieMCgYEA4db/SIrwN2+g1Gjo3oE09kd89VGjVRer
srbcqc7DcPXP6Lw42sx96h4jVWWqHVo3DfwFBdUb1LH2cnVXQjgDUHdNdpl01cf/
BFYCFINi2qKMqiJYswkhYxZ1BLz/zuQTDbPFL2PgLniKFG2aFLrTS3S/tgeB5QwI
RPigH3kfI6sCgYAPqsCJyFMlrvfRRNZdQewi4VnPsEPF4/hjpAs1gD8vfSoZWlkw
vylUd9HCerzgYaA7rixieQ0sxTvtxhL6PXlM2NEBFQbV16hPFL6/IiG4F0u9oHNo
eG8rHtqKlSjnBn4yoYFm70Dhe7QtbZelcaAoPCH6CUHj2St5B8ZHWDtREQKBgHNp
wER+XIy4C2UByCANv9csaXulIOdXlXNbaCGPfOm5dWrm5ddLMf33MO9vaSRe+ku3
Q4nbgsGLwPp1ZQZ+QZNZpMi7W6306yp4GdAJ5Pb+oww/ST0VqW5OB7dILyK4A9S4
zkiNrf+Rsl8GM/vsDhc9rsuDwqofIAq/VHVBHNzJAoGBAOHQof5L6iGHOHcxLazx
4MGvRTpmzU/PX6Q3QxqpetEGFEDZAaL58L67SSS3fFBnKrVAidF6llC1bAH1aoRa
fYHUDi45xBoroy0hBwrnTKRxppua4UK75FUH5PPJfR6cCvw5stRkzIevTZHhozkX
pM7PYH/x4BiBmgQ3bhOqTp4H
--estServerExampleBoundary
Content-Type: application/pkcs7-mime; smime-type=certs-only
Content-Transfer-Encoding: base64
```

```
MIIDRQYJKoZIhvcNAQcCoIIDNjCCAzICAQExADALBgkqhkiG9w0BBwGgggMYMIID
FDCCAfygAwIBAgIBFjANBgkqhkiG9w0BAQUFADAbMRkwFwYDVQQDExBlc3RFeGFt
cGxlQ0EgTndOMB4XDTEzMDUwOTIzMjU1NloXDTE0MDUwOTIzMjU1NlowLDEqMCgG
A1UEAxMhc2VydmVyc2lkZSBrZXkgZ2VuZXJhdGVkIHJlc3BvbnNlMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAz8CrcCe04zIAPj6I+uFfdPa8qpdH6D9Y
VOGKucvP2zj+WU3ugK4FYVMntPfQHPnX3cKcj/wxEoro1Bf3UutlXJN1FjW58PuD
t/BVKdHpy2240k7nePCqRFuvhc5HrgGpzxmTANcCsq2Mvaw+bUDjiVPwgVYVKyE8
mC2TWFHuGoG6XEdMDLa0a9hOfDLvMEPNK8pOp5C+wjpOWSEHTUcZXZOwxb6QPOWJ
9bLbpqBkbVS1udYl3k0tS7V8IblG/4+i3bN77wIidkomdGN68T5ZyVchZkfDH9kw
To87har2v1Sw9mskI4GhhI6tmVh13pUADmejd5awoaIxChRqvd8joQIDAQABo1Iw
UDAOBgNVHQ8BAf8EBAMCBLAwHQYDVR0OBBYEFKeZixu9F+appDX2SS5HaxmV6Jr4
MB8GA1UdIwQYMBaAFLHEaeZbowSn2Jejizu/uWqyMkI8MA0GCSqGSIb3DQEBBQUA
A4IBAQBHhLmRAKrnTapqqBObDM9IQDQPuwW+fW1gYwZKlSm/IWIwHEZL1igXhpjj
rf4xqpIkiJMmkaOeoXA8PFniX0/lZM9FQSM/j89CUf5dMoAqWj8s17xuXu9L/hVe
XjjXHsL40WuDG6tMPN9vcT8tE3ruor608MKSHFX/NEM6+AaNVSUPTmB33BgYB1Wa
E7pn3JMN6pjIxsHnF4pKi8qvoTSVVjaCEwUe8Q/fw1yvjoHoYJtyMn4v5Kb3Rt+m
s8Yie1tcfVQrjQutqr34/IJsKdPziZwi92KZa+1958A6M9O/p5OI0up9ZXKg2DEC
1O9qT0GyYJ6sxAyKiGTOxk6jMddDoQAxAA==
--estServerExampleBoundary--
This is the epilogue.  It is also to be ignored.
```

Appendix B. Contributors and 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 versions of this document. Our thanks also go the authors of [RFC6403], around whose document we structured part of this specification.

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