Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 25, 2009

A. Nourse
X. Liu
J. Vilhuber
C. Madson
Cisco Systems, Inc.

January 21, 2009

Cisco Systems' Simple Certificate Enrollment Protocol
draft-nourse-scep-18

Abstract

This document specifies the Simple Certificate Enrollment Protocol, a PKI communication protocol which leverages existing technology by using PKCS#7 and PKCS#10. SCEP is the evolution of the enrollment protocol developed by Verisign, Inc. for Cisco Systems, Inc. It now enjoys wide support in both client and CA implementations.

Table of Contents

1. Introduction

Public key technology is widely available and increasingly widely deployed. X.509 certificates serve as the basis for several standards-based security protocols in the IETF, such as IKE [RFC2409] and IKEv2 [RFC4306], and TLS [RFC4346]. When an X.509 certificate is issued by other than the certificate subject (a self-issued certificate), there typically is a need for a certificate management protocol. Such a protocol enables a PKI client to request a certificate, certificate renewal, or certificate revocation from a Certification Authority. Often there also is a need for protocols to request a certificate or cert status info, although these functions are often provided by distinct protocols, e.g., LDAP for X509 [RFC4523] or OCSP [RFC2560].

This specification defines a protocol, SCEP, for certificate management and certificate and CRL queries in a closed environment. While widely deployed, this protocol omits some certificate management features, e.g., in-band certificate revocation transactions, that can significantly enhance the security achieved in a PKI. The IETF protocol suite currently includes two certificate management protocols with more comprehensive functionality: CMP [RFC4210] and Certificate Management over CMS [RFC5272]. Where interoperability with the installed base of SCEP implementations is required, implementers are encouraged to support a comprehensive standards track certificate management protocol in addition to the protocol defined in this specification. This implementation strategy balances near term requirements for interoperability with longer term security goals.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. The Goal of SCEP

The goal of SCEP is to support the secure issuance of certificates to network devices in a scalable manner, using existing technology whenever possible. The protocol supports the following operations:

o CA and RA public key distribution

o Certificate enrollment

- Certificate query

- CRL query

Certificate and CRL access can be achieved by using the LDAP protocol (as specified in Appendix D), or by using the query messages defined in SCEP. The use of HTTP certificate and CRL access, and the support of CDP as specified in [RFC5280], is outside the scope of this document. In Section Section 2.1, we first define PKI entity types as well as the properties of each entity type. In Section Section 2.2, the PKI operations are described at functional level. Section Section 2.3 describes the transaction behavior of each PKI operations. The complete PKI messages are covered in Section Section 5.

2.1. SCEP Entity Types

The entity types defined in SCEP are the "requester" type (i.e., IPSEC clients), the Certificate Authority (CA) entity type, and the Registration Authority entity type (RA). A requester is sometimes called a "SCEP client" in the following.

2.1.1. Requesters

A requester is an entity whose name is defined in a certificate subject name field and optionally, in SubjectAltName, a X.509 certificate V3 extension. As a requester, a SCEP client is identified by a subject name consisting of the following naming attributes:

- Fully qualified domain name, for example, Router.cisco.com,

- IP Address,

- Serial number, and/or

- x.500 distinguished name.

The fully qualified domain name is required for a requester that intends to use the certificate with, for example, IKE [RFC2409]. The IP address, serial number, and x.500 distinguished name are optional name attributes.

In the certificate enrollment request, the PKCS#10 [RFC2986] subject field contains the required and optional name attributes. The distinguished name, if any, should be the subject name field, while any domain name, serial number, or IP address supplied should be in the subjectAltName field. The subjectName field may be empty (if

there is no distinguished name) or the subjectAltName may be omitted, but not both.

It is important to note that a client named as Alice.cisco.com is different than a client named as "Alice.cisco.com+192.0.2.4" (read Alice.cisco.com plus the IP address name attribute 192.0.2.4). From a CA point of view, the Distinguished names assigned in these two cases are distinct names.

Entity names which are specified as in the IPSEC profile (i.e., FQDN, IP address and User FQDN) must be presented in the certificate's SubjectAltName extension. Multiple IPSEC entity names, (if any) are encoded as multiple values of a single SubjectAltName extension. The CA has the authority to assign a distinguished name to a requester, whether or not one was included in the request. The assigned DN should contain the SCEP client names as the relative DN.

The attribute identifiers and an example of SCEP client subject name are specified in Appendix A. Appendix B has an example from Cisco VPN Client enrollment request.

2.1.1.1. Local Key/Certificate/CRL Storage and Certificate-name uniqueness

A requester is required to generate, and provide storage for, asymmetric keypairs. If the requester does not have enough permanent memory to save its certificate, then it should be able to query its own certificate from the CA or an LDAP server, once the certificate has been issued.

A keypair can be generated with a specific key usage. The key usage is conveyed to the CA through the certificate enrollment request. All current SCEP client implementations expect that there will be only one keypair for a given subjectName and key usage combination and CA, at any time. This property is called the certificate-name uniqueness property, and it implies that a CA that implements SCEP will enforce the unique mapping between a SCEP client subject name and its keypairs with a given key usage. At any time, if the subject name is changed, or if the keypair is updated, the existing certificate would have to be revoked before a new one could be issued.

It is desirable that the CA enforce certificate-name uniqueness, but it is not mandatory. However a CA that does not enforce uniqueness must provide some other mechanism to prevent the re-transmission of an enrollment request by a SCEP client from creating a second certificate or certificate request, nor can the second request merely be rejected. If a client times out from polling for a pending

request it can resynchronize by reissuing the original request with the original subject name, key, and transaction ID. This should return the status of the original transaction, including the certificate if it was granted. It should not create a new transaction unless the original certificate has been revoked, or the transaction arrives more than halfway through the validity time of the original certificate.

An enrollment request that occurs more than halfway through the validity time of an existing certificate for the same subjectName and key usage MAY be interpreted as a re-enrollment or renewal request and accepted. A new certificate with new validity dates may be issued, even though the old one is still valid, if the CA policy permits, as described in Section 2.1.1.3. See also Appendix G.

2.1.1.2. Requester authentication

As with every protocol that uses public-key cryptography, the association between the public keys used in the protocol and the identities with which they are associated must be authenticated in a cryptographically secure manner. This requirement is needed to prevent a "man in the middle" attack, in which an adversary can manipulate the data as it travels between the protocol participants and subvert the security of the protocol.

PKCS#10 [RFC2986] specifies the use of a ChallengePassword attribute to be sent as part of the enrollment request. SCEP uses this ChallengePassword to satisfy the above requirements for security. The PKCS#7 [RFC2315] envelope protects the privacy of the challenge password.

2.1.1.2.1. Manual enrollment authentication

In the manual mode, the requester is required to wait until its identity can be verified by the CA operator using any reliable out-of-band method. To prevent a "man-in-the-middle" attack, a SHA-1, SHA-256, SHA-512, or MD5 `fingerprint' generated on the PKCS#10 [RFC2986] (before PKCS#7 [RFC2315] enveloping and signing) SHOULD be compared out-of-band between the CA operator and the requester. SCEP clients and CAs (or RAs, if appropriate) MUST display this fingerprint to the operator to enable this verification if manual mode is used. Failing to provide this information leaves the protocol vulnerable to attack by sophisticated adversaries.

In this case the challenge password is only used to authenticate a request for the certificate's revocation.

2.1.1.2.2. Automated enrollment authentication

When utilizing a pre-shared secret scheme, the server should distribute a shared secret to the requester which can uniquely associate the enrollment request with the given end entity. The distribution of the secret must be private: only the end entity should know this secret. The actual binding mechanism between the requester and the secret is subject to the server policy and implementation.

When using the pre-shared secret scheme, the requester must enter the pre-distributed secret as the ChallengePassword. It can then also be used to authenticate a request for the certificate's revocation.

2.1.1.3. Requester Uses Existing CA-Issued or Self-Signed Certificates

In this protocol, the communication between the requester and the certificate authority is secured by using PKCS#7 [RFC2315] as the messaging protocol (see Section 4). PKCS#7 [RFC2315], however, is a data format which assumes the communicating entities already possess the peer's certificates and requires both parties use the issuer names and issuer assigned certificate serial numbers to identify the certificate in order to verify the signature and decrypt the message.

o If the requesting system already has a certificate issued by the CA, that certificate SHOULD be presented as credentials for the renewal of that certificate if the CA supports the "Renewal" capability and the CA policy permits the certificate to be renewed.

o If the requesting system has no certificate issued by the CA, but has credentials from a different CA, that certificate MAY be presented as credentials instead of a self-signed certificate. Policy settings on the SCEP server will determine if the request can be accepted or not.

o If the requester does not have an appropriate existing certificate, then a self signed certificate must be used instead. The self signed certificate MUST use the same subjectName as in the pkcs10 request.

During the certificate enrollment, the requester will attach the appropriate certificate to the signed certificate request.

When the Certificate Authority creates the PKCS#7 envelope on the issued certificate, it should use the public key, issuer name, and serial number conveyed in the above included certificate (from the SignerInfo section of the inner pkcs#7 of the request). This will

inform the end entity on which private key should be used to open the envelope. Note that when a client enrolls for separate encryption and signature certificates, it may use the signature certificate to sign both requests, and then expect its signature key to be used to encrypt both responses.

In any case, the recipientInfo on the envelope should reflect the key used to encrypt the request.

2.1.1.4. Trusted CA Store

To support interoperability between IPSEC peers whose certificates are issued by different CAs, SCEP allows the users to configure multiple trusted certificates. Trusted certificates are configured as such in the client, based on some out-of-band means such as a "fingerprint". These trusted certificates are used to verify certificate chains that end in those certificates.

2.1.2. Certificate Authority

A Certificate Authority(CA) is an entity whose name is defined in the certificate issuer name field. Before any PKI operations can begin, the CA generates its own public key pair and creates a self-signed CA certificate, or causes another CA to issue a certificate to it. Associated with the CA certificate is a fingerprint which will be used by the requester to authenticate the received CA certificate if it is self-signed. The fingerprint is created by calculating a SHA-1, SHA-256, SHA-512, or MD5 hash on the whole CA certificate. Before any requester can start its enrollment, this CA certificate has to be configured at the entity side securely. For IPSEC clients, the client certificates must have SubjectAltName extension. To utilize LDAP as a CRL query protocol, the certificates must have a CRL Distribution Point. Key usage is optional. Without key usage, the public key is assumed as a general purpose public key and it can be used for all purposes.

A Certificate Authority may enforce certain name policy. When using a X.500 directory name as the subject name, all the name attributes specified in the PKCS#10 [RFC2986] request should be included as Relative DN. All the name attributes as defined in [RFC5280] should be specified in the SubjectAltName. An example is provided in Appendix A.

If there is no LDAP or HTTP query protocol support, the Certificate Authority itself should answer certificate and CRL queries, and to this end it should be online all the time.

The updating of the CA's public key is addressed in Appendix G.

2.1.3. Registration Authorities

In an environment where an RA is present, a requester performs enrollment through the RA. In order to setup a secure channel with an RA using PKCS#7 [RFC2315], the RA certificate(s) have to be obtained by the client in addition to the CA certificate(s).

In the following, the CA and RA are specified as one entity in the context of PKI operation definitions.

2.2. SCEP Operations Overview

In this section, we give a high level overview of the PKI operations as defined in SCEP.

2.2.1. Requester Initialization

The requester initialization includes the key pair generation and the configuring of the required information to communicate with the certificate authority.

2.2.1.1. RSA Key Pairs

Before a requester can start a PKI transaction, it must have at least one RSA key pair.

Key pairs may be intended for particular purposes, such as encryption only, or signing only. The usage of any associated certificate can be restricted by adding key usage and extended key usage attributes to the PKCS#10 [RFC2986].

2.2.1.2. non-RSA Keys

SCEP does not support non-RSA keys. Though the protocol (being based on PKCS#7) does not preclude them, RSA is the only algorithm supported by current implementations.

2.2.1.3. Required Information

A requester is required to have the following information configured before starting any PKI operations:

1. the certificate authority IP address or fully-qualified domain name,

2. the certificate authority HTTP CGI script path,

3. the HTTP proxy information if there is no direct Internet connection to the server,

4. If CRLs are being published by the CA to an LDAP directory server, and there is a CRL Distribution Point containing only an X.500 directory name, then the client will need to know the LDAP server fully-qualified domain name or IP address. CRL Distribution Points are discussed in more detail in [RFC5280].

2.2.2. CA/RA Certificate Distribution

Before any PKI operation can be started, the requester needs to get the CA/RA certificates. At this time, since no public key has been exchanged between the requester and the CA/RA, the message to get the CA/RA certificate cannot be secured using PKCS#7 [RFC2315]. Instead, the CA/RA certificate distribution is implemented as a clear HTTP Get operation. After the requester gets the CA certificate, it has to authenticate the CA certificate by comparing the finger print with the CA/RA operator out-of-band. Since the RA certificates are signed by the CA, there is no need to authenticate the RA certificates.

This operation is defined as a transaction consisting of one HTTP Get message and one HTTP Response message:

```
          REQUESTER                           CA SERVER
   Get CA/RA Certificate: HTTP Get message
   ----------------------------->
                     CA/RA Certificate download: HTTP Response message
                              <------------------------------------------
   Compute finger print and
   call CA operator.
                                     Receive call and check finger print

                          Get CA/RA Certificate
```

If an RA is in use, a degenerated PKCS#7 [RFC2315] with a certificate chain consisting of both RA and CA certificates is sent back to the end entity. Otherwise the CA certificate is directly sent back as the HTTP response payload.

2.2.3. Certificate Enrollment

A requester starts an enrollment transaction by creating a certificate request using PKCS#10 [RFC2986] and sends it to the CA/RA enveloped using the PKCS#7 [RFC2315]. After the CA/RA receives the request, it will either automatically approve the request and send the certificate back, or it will require the requester to wait until the operator can manually authenticate the identity of the requester.

In the automatic mode, the transaction consists of one PKCSReq PKI Message, and one CertRep PKI message.

In the manual mode, the requester enters into polling mode by periodically sending a GetCertInitial PKI message to the server, until the server operator completes the manual authentication, after which the CA will respond to GetCertInitial by returning the issued certificate.

It is up to local CA policy (and CA implementation) as to whether a certificate is granted automatically, or whether it is manually granted by the administrator. The ChallengePassword MAY be used to automatically authenticate the request, but does not have to.

Polling mode is entered whenever the server returns a PENDING response.

The transaction in automatic mode:

```
         REQUESTER                            CA SERVER

     PKCSReq: PKI cert. enrollment msg
     --------------------------------> CertRep: pkiStatus = SUCCESS
                                       certificate attached
                                      <------------------------------
     Receive issued certificate.
```

The transaction in manual mode:

```
        REQUESTER                            CA SERVER
    PKCSReq: PKI cert. enrollment msg
    --------------------------------> CertRep: pkiStatus = PENDING
                                      <------------------------------
    GetCertInitial: polling msg
    --------------------------------> CertRep: pkiStatus = PENDING
                                      <------------------------------
    ................. <manual identity authentication...............

    GetCertInitial: polling msg
    --------------------------------> CertRep: pkiStatus = SUCCESS
                                      certificate attached
                                      <------------------------------
    Receive issued certificate.
```

2.2.4. Requester Certificate Revocation

SCEP currently only allows revocation as an out-of-band process. In order to revoke a certificate, the requester must contact the CA server operator, who MAY wish to verify the ChallengePassword (which has been sent to the server as an attribute of the PKCS#10 [RFC2986]

certificate request). If the ChallengePassword matches, the
certificate can be revoked.

2.2.5. **Certificate** Access

There are two methods to query certificates. The first method is to use LDAP as a query protocol. Using LDAP to query assumes the client understand the LDAP scheme supported by the CA. The SCEP client assumes that the subject DN name in the certificate is used as the URL to query the certificate. The standard attributes (userCertificate and caCertificate) are used as filter.

For the environment where LDAP is not available, a certificate query message is defined to retrieve the certificates from the CA.

To query a certificate from the certificate authority, a requester sends a request consisting of the certificate's issuer name and the serial number. This assumes that the requester has saved the issuer name and the serial number of the issued certificate from the previous enrollment transaction. The transaction to query a certificate consists of one GetCert PKI message and one CertRep PKI message:

```
        REQUESTER                            CA SERVER
     GetCert: PKI certificate query msg
     -------------------------------> CertRep:  pkiStatus = SUCCESS
                                      certificate attached
                                      <------------------------------
     Receive the certificate.
```

2.2.6. **CRL Distribution**

The CA/RA will not "push" the CRL to the end entities. The query of the CRL can only be initialized by the requester.

There are two methods to query CRL:

1. If the CA supports CRL Distribution Points [RFC5280] (section 4.2.1.13), then the CRL MUST be retrieved via the mechanism specified in the CDP. This is the preferred method. Please refer to Appendix D for the examples of CRL Distribution Point.

2. If the CA does not support CDP's, a CRL query is composed by creating a message consisting of the CA issuer name and the CA's certificate serial number. This method is deprecated because it does not scale well and requires the CA to be a high-availability service.

The message is sent to the CA in the same way as the other SCEP

requests: The transaction to query CRL consists of one GetCRL PKI message and one CertRep PKI message which contain only the CRL (no certificates).

```
         REQUESTER                            CA SERVER
     GetCRL: PKI CRL query msg
  ----------------------------------->
                                    CertRep:  CRL attached
                                 <--------------------------------
```

2.3. PKI Operation Transactional Behavior

As described before, a PKI operation is a transaction consisting of the messages exchanged between a requester and the CA/RA. This section will specify the transaction behavior on both the requester and the certificate authority server. Because the protocol is basically a two way communication protocol without a confirmation message from the initiating side, state and state resynchronization rules have to be defined, in case any error happens at either side. Before the state transition can be defined, the notion of transaction identifier has to be defined first.

2.3.1. Transaction Identifier

A transaction identifier is a string generated by the entity when starting a transaction. Since all the PKI operations defined in this protocol are initiated by the requester, it is the responsibility of the requester to generate a unique string as the transaction identifier. All the PKI messages exchanged for a given PKI transaction must carry the same transaction identifier.

The transaction identifier is generated as a SHA-1, SHA-256, SHA-512 or MD5 hash on the public key value for which the enrollment request is made. This allows the SCEP client to reuse the same transaction identifier if it is reissuing a request for the same certificate (i.e. a certificate with the same subject, issuer, and key). The SCEP protocol requires that transaction identifiers be unique, so that queries can be matched up with transactions. For this reason, in those cases in which separate signing and encryption certificates are issued to the same requester, the keys must be different.

2.3.2. State Transitions in Certificate Enrollment

The requester state transitions during enrollment operation are indicated in the diagram below:

```
                                  +-<------+
                                  |        |
                                 GetCertInitial triggered by timeout or
                                  |        |    manual authentication
                                  |        |
   [CERT-NONEXISTANT] ------> [CERT-REQ-PENDING] ---> [CERT-ISSUED]
         |           PKCSReq          |        CertRep with SUCCESS
         |                            |
         |                            |
         +--------<-------------------+
         request rejected, timeout, or error
```

As described in the section 2.2.3, certificate enrollment starts at the state CERT-NONEXISTANT. Sending PKCSReq changes the state to CERT-REQ-PENDING. Receiving CertRep with SUCCESS status changes the state to CERT-ISSUED. In the case the server sending back the response with pending status, the requester will keep polling certificate response by sending GetCertInitial to the server, until either a CertRep with SUCCESS status is received, or the maximum polling number has been exceeded.

If an error or timeout occurs in the CERT-REQ-PENDING state, the end entity will transition to the CERT-NONEXISTANT state.

The client administrator will, eventually, start up another enrollment request. It is important to note that, as long as the requester does not change its subject name or keys, the same transaction id will be used in the "new" transaction. This is important because based on this transaction id, the certificate authority server can recognize this as an existing transaction instead of a new one.

2.3.3. Transaction Behavior of Certificate/CRL Access

There is no state maintained during certificate access and CRL access transaction. When using the certificate query and CRL query messages defined in this protocol, the transaction identifier is still required so that the requester can match the response message with the upstanding request message. When using LDAP to query the certificate and the CRL, the behavior is specified by the LDAP protocol.

2.4. Security

The security goals of SCEP are that no adversary can:

- o subvert the public key/identity binding from that intended,
- o discover the identity information in the enrollment requests and issued certificates,
- o cause the revocation of certificates with any non-negligible probability.

Here an adversary is any entity other than the requester and the CA (and optionally the RA) participating in the protocol that is computationally limited, but that can manipulate data during transmission (that is, a man-in-the-middle). The precise meaning of 'computationally limited' depends on the implementer's choice of cryptographic hash functions and ciphers. The required algorithms are RSA, DES and MD5. Depending on the CA Capabilities, Triple-DES may be used instead of DES, and SHA-1, SHA-256, or SHA-512 may be used instead of MD5. [See Appendix F].

The first and second goals are met through the use of PKCS#7 [RFC2315] and PKCS#10 [RFC2986] encryption and digital signatures using authenticated public keys. The CA's public key is authenticated via the checking of the CA fingerprint, as specified in Section 2.1.2, and the SCEP client's public key is authenticated through the manual authentication or pre-shared secret authentication, as specified in Section 2.1.1.2. The third goal is met through the use of a Challenge Password for revocation, that is chosen by the SCEP client and communicated to the CA protected by the PKCS#7 [RFC2315] encryption, as specified in Section 2.2.4.

The motivation of the first security goal is straightforward. The motivation for the second security goal is to protect the identity information in the enrollment requests and certificates. For example, two IPSEC hosts behind a firewall may need to exchange certificates, and may need to enroll certificates with a CA that is outside of a firewall.

Most networks with firewalls seek to prevent IP addresses and DNS information from the trusted network leaving that network. The second goal enables the hosts in this example to enroll with a CA outside the firewall without revealing this information. The motivation for the third security goal is to protect the SCEP clients from denial of service attacks.

3. Transport Protocol

In the SCEP protocol, HTTP is used as the transport protocol for the PKI messages.

3.1. HTTP "GET" and "POST" Message Format

The following is the syntax definition of a HTTP GET message sent from a requester to a certificate authority server:

```
"GET " CGI-PATH CGI-PROG "?operation=" OPERATION "&message=" MESSAGE
```

where:

- o CGI-PATH defines the actual CGI path to invoke the CGI program which parses the request.

- o CGI-PROG is set to be the string "pkiclient.exe". This is intended to be the program that the CA will use to handle the SCEP transactions, though the CA may ignore CGI-PROG and use only the CGI-PATH.

- o OPERATION is set to be the string

- o

  * "PKIOperation" when the GET message carries a PKI message to request certificates or CRL

  * "GetCACaps", "GetCACert", "GetNextCACert" or "GetCACertChain" when the GET operation is used to get CA capabilities, CA/RA certificate, the replacement CA/RA certificates for when the current ones expire, or the CA Certificate chain (respectively)

- o MESSAGE is

- o

  * a base64-encoded PKI message , when OPERATION is "PKIOperation" and method is GET.

  * a CRL distribution point in URI format , when OPERATION is GetCRL,

  * a string which represents the certificate authority issuer identifier otherwise.

SCEP uses the HTTP "GET" and "POST" messages to request information from the CA. Requests for CA certificates or capabilities are sent in the clear, using "GET", with the OPERATION and MESSAGE fields identifying the requested data. CRLs may also be requested in the clear if the CA supports it.

Other types of requests are sent using the PKCS#7 [RFC2315] data format. These may be issued by means of a GET operation with OPERATION and MESSAGE parameters in the Request-URL. OPERATION identifies the type of GET operation, and MESSAGE is actually the PKCS#7 [RFC2315] message Base64-Encoded.

For example. a requester may submit a message via HTTP to the server as follows:

```
GET /cgi-bin/pkiclient.exe?operation=PKIOperation&message=MIAGCSqGSIb3D
QEHA6CAMIACAQAxgDCBzAIBADB2MGIxETAPBgNVBAcTCE ......AAAAAA== HTTP/1.0
```

If supported by the CA, the message may also be sent via HTTP POST:

```
POST /cgi-bin/pkiclient.exe?operation=PKIOperation HTTP/1.0
Content-Length: 1234

<binary pkcs7 data>
```

This is further described in Appendix H. To determine if the CA supports POST, use the GetCACaps message described in Appendix F.

3.2. Response Message Format

For each GET operation, the CA/RA server will return a MIME object via HTTP. For a GET operation with PKIOperation as its type, the response is tagged as having a Content Type of application/x-pki-message. The body of this message is a BER encoded binary PKI message. The following is an example of the response:
"Content-Type:application/x-pki-message\n\n"<BER-encoded PKI msg>

In the case of GET operation with a type of GetCACert the MIME content type returned will depend on whether or not an RA is in use. If there is no RA, only the CA certificate is sent back in the response, and the response has the content type tagged as application/x-x509-ca-cert. The body of the response is a DER encoded binary X.509 certificate. For example:
"Content-Type:application/x-x509-ca-cert\n\n"<BER-encoded X509>

If there is an RA, the RA certificates are sent back together with the CA certificates, a certificate-only PKCS#7 [RFC2315] SignedData is sent back in the response where the SignerInfo is empty. Section 5 has the detailed definition of the message format in this case. The content type is application/x-x509-ca-ra-cert.

The response to GetNextCACert is always a certificates-only PKCS#7 [RFC2315] SignedData with a content type of application/x-x509-ca-ra-cert. If there is an RA, The signer is the current RA certificate. Otherwise, the signer is the current CA certificate.

If the CA supports it, PKIOperation may also be done via an HTTP POST. This is described in Appendix H.

4. Secure Transportation: PKCS#7

PKCS#7 [RFC2315] is a general enveloping mechanism that enables both signed and encrypted transmission of arbitrary data. It is widely implemented and included in the RSA tool kit. In this section, the general PKCS#7 [RFC2315] enveloped PKI message format is specified.

All messages MUST be valid PKCS#7 [RFC2315] structures, unless otherwise noted.

4.1. SCEP Message Format

An SCEP message consists of an information portion (which depends on the type of SCEP message being sent) and a set of transaction-specific Attributes (see section Section 4.2).

The message-specific information is used as EnvelopeContent in an SCEP pkcsPKIEnvelope message (see section Section 4.1.1).

Next, the pkcsPKIEnvelope is used as Content for a pkiMessage SCEP type (see section Section 4.1.2).

The transaction-specific attributes are encoded as a set of authenticatedAttributes in the SignerInfo of the SignedData.

By applying both enveloping and signing transformations, a SCEP message is protected both for the integrity of its end-end-transition information and the confidentiality of its information portion. The advantage of this technique over the conventional transaction message format is that, the signed transaction type information and the status of the transaction can be determined prior to invoke security handling procedures specific to the information portion being processed.

4.1.1. SCEP pkcsPKIEnvelope

The SCEP messages are carried inside an Enveloped-data content type, as defined in PKCS#7 Section 10, with the following restrictions:

o version shall be 0

o EncryptedContent shall be the SCEP message being transported (see section Section 5)

The message is encrypted using the public key of the recipient, i.e. the RA or the CA the message is for.

NOTE: The PKCS#7 [RFC2315] EncryptedContent is specified as an octet string, but SCEP entities must also accept a sequence of octet strings as a valid alternate encoding.

4.1.2. SCEP pkiMessage type

A SCEP pkiMessage consists of an Signed-data content type, as defined in PKCS#7 Section 9. The following restrictions apply:

o version shall be 1

o The signed content shall be a pkcsPKIEnvelope

o The SignerInfo MUST contain a set of authenticatedAttributes (see PKCS#7 Section 9.2 as well as section Section 4.2 in this document. All messages are required to contain a SCEP transactionID attribute, an SCEP messageType, and, of course, any attributes required by PKCS#7 section 9.2. It may have other attributes as well, depending on the messageType.

The message is signed in one of two ways: The requester can generate a self-signed certificate, or the requester can use a previously issued certificate, if the RA/CA supports the RENEWAL option.

4.2. Signed Transaction Attributes

The following transaction attributes are encoded as authenticated attributes, and are carried, as specified in PKCS#7 Section 9.2, in the SignerInfo for this signedData.

Please refer to Appendix B for the OID definitions.

Attribute	Encoding	Comment
transactionID	PrintableString	Decimal value as a string
messageType	PrintableString	Decimal value as a string
pkiStatus	PrintableString	Decimal value as a string
failInfo	PrintableString	Decimal value as a string
senderNonce	OctetString	
recipientNonce	OctetString	

The attributes are detailed in the following sections.

4.2.1. transactionID

The transactionID is an attribute which uniquely identifies a transaction. This attribute is required in all PKI messages.

Because the enrollment transaction could be interrupted by various errors, including network connection errors or client reboot, the SCEP client generates a transaction identifier by calculating a hash on the public key value for which the enrollment is requested. This retains the same transaction identifier throughout the enrollment transaction, even if the client has rebooted or timed out, and issues a new enrollment request for the same key pair.

It also provides the way for the CA to uniquely identify a transaction in its database. At the requester side, it generates a transaction identifier which is included in PKCSReq. If the CA returns a response of PENDING, the requester will poll by periodically sending out GetCertInitial with the same transaction identifier until either a response other than PENDING is obtained, or the configured maximum time has elapsed.

For non-enrollment message (for example GetCert and GetCRL), the transactionID should be a number unique to the client.

4.2.2. messageType

The messageType attribute specify the type of operation performed by the transaction. This attribute is required in all PKI messages. Currently, the following message types are defined:

o PKCSReq (19) -- PKCS#10 [RFC2986] certificate request

o CertRep (3) -- Response to certificate or CRL request

o GetCertInitial (20) -- Certificate polling in manual enrollment

o GetCert (21) -- Retrieve a certificate

o GetCRL (22) -- Retrieve a CRL

4.2.3. pkiStatus

All response message will include transaction status information which is defined as pkiStatus attribute:

o SUCCESS (0) -- request granted

- o FAILURE (2) -- request rejected. This also requires a failInfo attribute to be present, as defined in section 4.2.4.

- o PENDING (3) -- request pending for manual approval

4.2.4. failInfo

The failInfo attribute will contain one of the following failure reasons:

- o badAlg (0) -- Unrecognized or unsupported algorithm ident

- o badMessageCheck (1) -- integrity check failed

- o badRequest (2) -- transaction not permitted or supported

- o badTime (3) -- Message time field was not sufficiently close to the system time

- o badCertId (4) -- No certificate could be identified matching the provided criteria

4.2.5. senderNonce and responderNonce

The attributes of senderNonce and recipientNonce are the 16 byte random numbers generated for each transaction to prevent the replay attack.

When a requester sends a PKI message to the server, a senderNonce is included in the message. After the server processes the request, it will send back the requester senderNonce as the recipientNonce and generates another nonce as the senderNonce in the response message. Because the proposed PKI protocol is a two-way communication protocol, it is clear that the nonce can only be used by the requester to prevent the replay. The server has to employ extra state related information to prevent a replay attack.

5. SCEP Transaction Specification

In this section each SCEP transaction is specified in terms of the complete messages exchanged during the transaction.

5.1. Certificate Enrollment

The certificate enrollment transaction consists of one PKCSReq message sent to the certificate authority from a requester, and one CertRep message sent back from the server.

Precondition: Both the requester and the certificate authority have completed their initialization process. The requester has already been configured with the CA/RA certificate.

Postcondition: Either the certificate is received by the requester, or the end entity is notified to do the manual authentication, or the request is rejected.

5.1.1. PKCSReq Message Format

A PKCSReq message is a pkiMessage (see section Section 4.1.2), with the messageType attribute set to PKCSReq.

The information portion of the pkiMessage is a PKCS#10 [RFC2986] certificate request, which MUST contain at least the following items:

- o the subject Distinguished Name
- o the subject public key
- o a ChallengePassword attribute. The Challenge Password may be used to (out-of-band) authenticate the enrollment request itself, or in an out-of-band revocation request for the issued certificate.

Of course the certificate request may also contain any additional fields that make up a valid PKCS#10 request, including but not limited to an ExtensionReq attribute which is a sequence of extensions the requester expects to be included in its V3 certificate extensions (for example key-usage and extended key-usages).

This pkiMessage of type PKCSReq MUST contain:

- o A transactionID attribute, calculated as per section Section 4.2.1
- o a messageType attribute set to PKCSReq
- o a senderNonce, calculated as per section Section 4.2.5

The pkiMessage is then encrypted into a pkcsPKIEnvelope message (see section Section 4.1.1).

5.1.2. CertRep Message Format

The response to an SCEP enrollment request (PKCSReq) is a CertRep message.

If the request is granted, the pkiStatus is set to SUCCESS, and the certificate is returned.

If the request is rejected, the pkiStatus is set to FAILURE, and a failInfo attribute is returned.

If the server requires manual approval of the request, the pkiStatus is set to PENDING.

5.1.2.1. PENDING Response

When the CA is configured to manually authenticate the requester, the CertRep is returned with the attribute pkiStatus set to PENDING. The data portion for this message is null. In addition to the attributes required by PKCS#7, the following SCEP attributes are required:

- messageType set to CertReq
- transactionID copied from the PKCSReq message
- pkiStatus set to PENDING
- senderNonce as defined in section Section 4.2.5
- recipientNonce as defined in section Section 4.2.5

5.1.2.2. FAILURE Response

In this case, the CertRep sent back to the requester is same as in the PENDING case, except that the pkiStatus attribute is set to FAILURE, and the failInfo attribute should be included.

5.1.2.3. SUCCESS response

In this case, the information portion of CertRep will be a degenerated PKCS#7 [RFC2315] which contains the requester's newly generated certificate. The CertRep is otherwise identical to the PENDING reply, with the exception that pkiStatus is set to SUCCESS.

5.2. Poll for Requester Initial Certificate

Either triggered by the PENDING status received from the CertRep, or by the non-response timeout for the previous PKCSReq, a requester will enter the polling state by periodically sending GetCertInitial to the server, until either the request is granted and the certificate is sent back, or the request is rejected, or the configured time limit for polling is exceeded.

Since GetCertInitial is part of the enrollment, the messages exchanged during the polling period should carry the same transactionID attribute as the previous PKCSReq.

PreCondition: Either the requester has received a CertRep with pkiStatus set to PENDING, or waiting for a previous PKCSReq has timed out.

PostCondition: The requester has either received a valid response, which could be either a valid certificate (pkiStatus == SUCCESS), or a FAILURE message, or the polling period times out.

5.2.1. GetCertInitial Message Format

Since at this time the certificate has not been issued, the requester can only use the requester's subject name (which was contained in the original PKCS#10 sent via PKCSReq), combined with the transactionID attribute, to identify the polled certificate request.

The certificate authority server must be able to uniquely identify the polled certificate request. A subject name can have more than one outstanding certificate request (with different key usage attributes).

The content of the pkiMessage with messageType GetCertInitial is an ASN.1 issuerAndSubject type (see figure Figure 1).

In addition to the authenticatedAttributes required for a valid PKCS#7, the pkiMessage MUST include the following:

- messageType set to GetCertInitial

- transactionID

- senderNonce

```
issuerAndSubject
issuerAndSubject ::= SEQUENCE {
    issuer Name,
    subject Name
}
```

Figure 1

5.2.2. GetCertInitial Response Message Format

The response messages for GetCertInitial are the same as for PKCSReq.

5.3. Certificate Access

The certificate query message defined in this section is an option when the LDAP server is not available to provide the certificate

query. A requester should be able to query an issued certificate from the certificate authority, as long as the issuer name and the issuer assigned certificate serial number is known to the requesting end entity. This transaction is not intended to provide the service as a certificate directory service. A more complicated query mechanism would have to be defined in order to allow a requester to query a certificate using various different fields.

This transaction consists of one GetCert message sent to the server by a requester, and one CertRep message sent back from the server.

PreCondition: The queried certificate have been issued by the certificate authority and the issuer assigned serial number is known.

PostCondition: Either the certificate is sent back or the request is rejected.

5.3.1. GetCert Message Format

The queried certificate is identified by its issuer name and the issuer assigned serial number. If this is a query for an arbitrary requester's certificate, the requesting requester should includes its own CA issued certificate in the signed envelope. If this is a query for its own certificate (assume the requester lost the issued certificate, or does not have enough non-volatile memory to save the certificate), then the self-signed certificate has to be included in the signed envelope.

The content of the pkiMessage with messageType GetCert is an ASN.1 IssuerAndSerial type, as specified in PKCS#7 Section 6.7.

In addition to the authenticatedAttributes required for a valid PKCS#7, the pkiMessage MUST include the following:

o messageType set to GetCert

o transactionID

o senderNonce

5.3.2. CertRep Message Format

In this case, the CertRep from the server is same as the CertRep for the PKCSReq, except that the server will only either grant the request (SUCCESS) or reject the request (FAILURE).

Also, the recipientInfo should use the CA issuer name and CA assigned serial number to identify the requester's key pair since at this

time, the requester has received its own certificate.

5.4. CRL Access

The CRL query message defined in this section is an option when the LDAP server is not available to provide the CRL query. In the PKI protocol proposed here, only the requester can initiate the transaction to download CRL.

A requester sends GetCRL request to the server and the server sends back CertRep whose information portion is a degenerated PKCS#7 [RFC2315] which contains only the most recent CRL. The size of CRL included in the CertRep should be determined by the implementation.

When a CRLDistributionPoint is used, the GetCRL exchange should not be used. Instead, the CRLDistributionPoint, as set in the certificate, should be queried (see [RFC5280] section 4.2.1.14).

PreCondition: The certificate authority certificate has been downloaded to the end entity.

PostCondition: CRL sent back to the requester.

5.4.1. GetCRL Message format

The CRL is identified by using both CA's issuer name and the CA certificate's serial number, combined into an ASN.1 issuerAndSerialNumber type, as defined in PKCS#7 Section 6.7. This constitutes the content of the pkiMessage with messageType GetCRL.

In addition to the authenticatedAttributes required for a valid PKCS#7, the pkiMessage MUST include the following:

o messageType set to GetCRL

o transactionID

o senderNonce

5.4.2. CertRep Message Format

The CRL is sent back to the requester through CertRep message. The information portion of this message is a degenerated PKCS#7 [RFC2315] SignedData which contains only the raw DER encoded CRL.

5.5. Get Certificate Authority Certificate

To get the CA certificate, the requester does a "HTTP GET" with a URL that identifies a CGI script on the server and an optional CA issuer identifier as the parameter to the CGI script.

The response is either a single X.509 CA certificate ("CA mode"), or a PKCS7 message containing the CA certificate and RA certificates ("RA mode"). The client can determine which mode the CA operates in by which response it gets.

Once the CA certificate is received by the requester, a fingerprint is generated using the SHA1, SHA256, SHA512 or MD5 hash algorithm on the whole CA certificate. If the requester does not have a certificate path to a trusted CA certificate, this fingerprint may be used to verify the certificate, by some positive out-of-band means, such as a phone call.

5.5.1. GetCACert HTTP Message Format

```
"GET" CGI-PATH CGI-PROG "?operation=GetCACert" "&message=" CA-IDENT
```

where:

- o CGI-PATH defines the actual CGI path to invoke the CGI program which parses the request.
- o CGI-PROG is set to be the string "pkiclient.exe" and this is expected to be the program that the CA will use to handle the SCEP transactions.
- o CA-IDENT is any string which is understood by the CA. For example, it could be a domain name like ietf.org. If a certificate authority has multiple CA certificates this field can be used to distinguish which is required. Otherwise it may be ignored.

5.5.2. Response

The response for GetCACert is different between the case where the CA directly communicated with the requester during the enrollment, and the case where a RA exists and the requester communicates with the RA during the enrollment.

5.5.2.1. CA Certificate Only Response

A binary X.509 CA certificate is sent back as a MIME object with a Content-Type of application/x-x509-ca-cert.

5.5.2.2. CA and RA Certificates Response

When an RA exists, both CA and RA certificates must be sent back in the response to the GetCACert request. The RA certificate(s) must be signed by the CA. A certificates-only PKCS#7 [RFC2315] SignedData is used to carry the certificates to the requester, with a Content-Type of application/x-x509-ca-ra-cert.

5.5.3. Get Next Certificate Authority Certificate

5.5.3.1. GetNextCACert HTTP Message Format

```
"GET" CGI-PATH CGI-PROG "?operation=GetNextCACert" "&message=" CA-
IDENT
```

The response to this message is a PKCS#7 [RFC2315] certificates-only message containing a CA certificate (and possibly RA certificates) to be used when the current CA certificate expires, signed with the current CA certificate (or RA certificate, if the CA is in RA mode. Note that a PKCS#7 [RFC2315] is returned even in CA mode.

5.5.3.2. GetCACaps HTTP Message Format

```
"GET" CGI-PATH CGI-PROG "?operation=GetCACaps" "&message=" CA-IDENT
```

This message requests capabilities from CA. The response is a list of text capabilities, as defined in Appendix F. Support for this message is optional, but if it is not supported, the client should assume that none of the capabilities in Appendix F are supported.

5.6. Get Certificate Authority Certificate Chain

GetCACertChain provides a way to get the entire certificate chain.

5.6.1. GetCACertChain HTTP Message Format

```
"GET" CGI-SCRIPT "?" "operation=GetCACertChain" "&" "message" CA-
IDENT
```

where:

CGI-SCRIPT and CA-IDENT are as described for GetCACert.

5.6.2. Response

The response for GetCACertChain is a certificates-only PKCS#7 [RFC2315] SignedData to carry the certificates to the requester, with a Content-Type of application/x-x509-ca-ra-cert-chain.

5.6.3. Backwards Compatibility

Versions of SCEP prior to revision 3 do not support GetCACertChain or GetNextCACert. Certificate Authorities written to these prior versions will not be able to process the message and may return an HTML error.

To avoid this, clients should send the GetCACert message first. If the returned certificate is self-signed or is signed by a Certificate Authority that is trusted by the client, then it is not necessary to send the GetCACertChain message and it should not be sent.

An old CA in a two-deep hierarchy might still get this message from a client if the client did not trust either that CA or its issuer.

6. Acknowledgements

The authors would like to thank Peter William of ValiCert, Inc. (formerly of Verisign, Inc) and Alex Deacon of Verisign, Inc. and Christopher Welles of IRE, Inc. for their contributions to this protocol and to this document.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

8.1. General Security

Common key-management considerations such as keeping private keys truly private and using adequate lengths for symmetric and asymmetric keys must be followed in order to maintain the security of this protocol.

This is especially true for CA keys, which, when compromised, compromise the security of all relying parties.

8.2. Use of the CA keypair

A CA keypair is generally meant for (and is usually flagged as) "certificate signing" (exclusively), rather than 'data signing' or 'data encryption'. The SCEP protocol, however, uses the CA keypair indiscriminately to encrypt and sign PKCS#7 transport messages. This is generally considered undesirable, as this weakens the CA keypair

over time (it is generally accepted that using any key weakens it over time, as it gives an attacker more data to work with).

While the CA keypair can be generated with the 'data encryption' and 'data signing' flags set, this is operationally not encouraged. It would make using the key as a PKCS#7 transport key 'legal', but the discussion from the previous paragraph still applies.

A solution is to use RA keys to secure the SCEP transport (i.e. message signing and encrypting), which allows the CA keys to be used only for their intended purpose of "certificate signing".

An RA can be implemented in two ways: physically separate or implicit. In the implicit case, the CA simply creates an extra keypair. A physically separate RA allows the CA to be inside the secure network, not accessible to hackers at all.

8.3. ChallengePassword

The ChallengePassword sent in the PKCS#10 enrollment request is signed (via inclusion in a pkiMessage) and encrypted (via inclusion in a pkcsPKIEnvelope). When saved by the CA, care should be taken to protect this password.

If the ChallengePassword is used to automatically authenticate an enrollment request, it is recommend that some form of one-time password be used to minimize damage in the event the data is compromised.

8.4. transactionID

A well-written CA/RA will not rely on the transactionID to be correct or as specified in this document. Requesters with buggy software might add additional undetected duplicate requests to the CA's queue (or worse). A well-written CA/RA should never assume the data from a requester is well-formed.

8.5. Unnecessary cryptography

Some of the SCEP exchanges use signing and encryption operations that are not necessary. In particular the GetCert and GetCRL exchanges are encrypted and signed (in both directions), where simply signing the request would suffice (CRL's and Certificates, i.e. the respective responses, are already signed by the CA and can be verified by the recipient).

This may affect performance and scalability of the CA which could be used as an attack vector on the CA (though not an anonymous one).

The use of CDP's is recommended for CRL access, as well as other ways of retrieving certificates (LDAP, direct HTTP access, etc).

9. Intellectual Property

This protocol includes the optional use of Certificate Revocation List Distribution Point (CRLDP) technology, which is a patented technology of Entrust Technologies, Inc. (Method for Efficient Management of Certificate Revocation Lists and Update Information (U.S. Patent 5,699,431)). Please contact Entrust Technologies, Inc. (www.entrust.com) for more information on licensing CRLDP technology.

10. Normative References

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

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4523] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates", RFC 4523, June 2006.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS

              (CMC)", RFC 5272, 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.

Appendix A. Cisco Requester Subject Name Definition

The ip address and the FQDN of a SCEP client should be included in the V3 extension subjectAltName. When the subjectAltName extension attribute is present, both the subjectAltName fields and the subjectName field could have the IP address and the FQDN information.

When the X.500 directory is used by the CA to define the name space, the subject name defined above become a RDN which is part of DN bound to the requester's public key in the certificate. A sample of DN assigned by Entrust CA is given below (assume the same ciscoRouterAlice is used as the requester defined subject name):
OU = InteropTesting, O = Entrust Technologies, C = CA
RDN = {"alice.cisco.com", "172.21.114.67", "22334455"}

Appendix B. IPSEC Client Enrollment Certificate Request

The following is the certificate enrollment request (PKCS#10 [RFC2986]) as created by Cisco VPN Client:

```
-----END NEW CERTIFICATE REQUEST-----
   0 30  439: SEQUENCE {
   4 30  288:   SEQUENCE {
   8 02    1:     INTEGER 0
  11 30   57:     SEQUENCE {
  13 31   55:       SET {
  15 30   53:         SEQUENCE {
  17 06    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
  22 13   46:           PrintableString
            :             'For Xiaoyi, IPSEC attrs in alternate name
              extn'
            :           }
            :         }
            :       }
  70 30  158:     SEQUENCE {
  73 30   13:       SEQUENCE {
  75 06    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1
                                                      1 1)
  86 05    0:         NULL
            :         }
```

```
 88 03  140:       BIT STRING 0 unused bits
           :         30 81 88 02 81 80 73 DB 1D D5 65 AA EF C7 D4 8E
           :         AA 6E EB 46 AC 91 2A 0F 50 51 17 AD 50 A2 2A F2
           :         CE BE F1 E4 22 8C D7 61 A1 6C 87 61 62 92 CB A6
           :         80 EA B4 0F 09 9D 18 5F 39 A3 02 0E DB 38 4C E4
           :         8A 63 2E 72 8B DC BE 9E ED 6C 1A 47 DE 13 1B 0F
           :         83 29 4D 3E 08 86 FF 08 2B 43 09 EF 67 A7 6B EA
           :         77 62 30 35 4D A9 0F 0F DF CC 44 F5 4D 2C 2E 19
           :         E8 63 94 AC 84 A4 D0 01 E1 E3 97 16 CD 86 64 18
           :                 [ Another 11 bytes skipped ]
           :       }
231 A0   63:     [0] {
233 30   61:       SEQUENCE {
235 06    9:         OBJECT IDENTIFIER extensionReq (1 2 840 113549 1 9
                                                  14)
246 31   48:         SET {
248 30   46:           SEQUENCE {
250 30   44:             SEQUENCE {
252 06    3:               OBJECT IDENTIFIER subjectAltName (2 5 29 17)
257 04   37:               OCTET STRING
                             30 23 87 04 01 02 03 04 81 0D 65 6D 61 69


                             6C 40 69 72 65 2E 63 6F 6D 82 0C 66 71 64
                             6E 2E 69 72 65 2E 63 6F 6D
           :               }
           :             }
           :           }
           :         }
           :       }
           :     }

296 30   13:   SEQUENCE {
298 06    9:     OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549
                                                        1 1 4)
309 05    0:     NULL
           :     }
311 03  129:   BIT STRING 0 unused bits
           :     19 60 55 45 7F 72 FD 4E E5 3F D2 66 B0 77 13 9A
           :     87 86 75 6A E1 36 C6 B6 21 71 68 BD 96 F0 B4 60
           :     95 8F 12 F1 65 33 16 FD 46 8A 63 19 90 40 B4 B7
           :     2C B5 AC 63 17 50 28 F0 CD A4 F0 00 4E D2 DE 6D
           :     C3 4F F5 CB 03 4D C8 D8 31 5A 7C 01 47 D2 2B 91
           :     B5 48 55 C8 A7 0B DD 45 D3 4A 8D 94 04 3A 6C B0
           :     A7 1D 64 74 AB 8A F7 FF 82 C7 22 0A 2A 95 FB 24
           :     88 AA B6 27 83 C1 EC 5E A0 BA 0C BA 2E 6D 50 C7
           :   }
```

Appendix C. Private OID Definitions

The OIDs used in SCEP are VeriSign self-maintained OIDs.

Name	ASN.1 Definition
id-VeriSign	OBJECT_IDENTIFIER ::= {2 16 US(840) 1 VeriSign(113733)}
id-pki	OBJECT_IDENTIFIER ::= {id-VeriSign pki(1)}
id-attributes	OBJECT_IDENTIFIER ::= {id-pki attributes(9)}
id-messageType	OBJECT_IDENTIFIER ::= {id-attributes messageType(2)}
id-pkiStatus	OBJECT_IDENTIFIER ::= {id-attributes pkiStatus(3)}
id-failInfo	OBJECT_IDENTIFIER ::= {id-attributes failInfo(4)}
id-senderNonce	OBJECT_IDENTIFIER ::= {id-attributes senderNonce(5)}
id-recipientNonce	OBJECT_IDENTIFIER ::= {id-attributes recipientNonce(6)}
id-transId	OBJECT_IDENTIFIER ::= {id-attributes transId(7)}
id-extensionReq	OBJECT_IDENTIFIER ::= {id-attributes extensionReq(8)}

Appendix D. CRL Query by means of LDAP

In order to retrieve the CRL by means of LDAP, the client needs to know where in the directory it is stored. The certificate must contain a CRL Distribution Point extension encoded as a DN or as an LDAP URI.

For example, the certificate issued by Entrust VPN contains the following DN as the CRL distribution point:
CN = CRL1, O = cisco, C = US

The ASN.1 encoding of this distribution point is:

```
30 2C 31 0B 30 09 06 03 55 04 06 13 02 55 53 31 0E 30 0C 06
03 55 04 0A 13 05 63 69 73 63 6F 31 0D 30 0B 06 03 55 04 03
13 04 43 52 4C 31
```

The ldap form would be:
ldap://servername/CN=CRL1,O=cisco,C=US

Appendix E. SCEP State Transitions

SCEP state transitions are indexed by the transactionID attribute. The design goal is to ensure the synchronization between the CA and the requester under various error situations. An identity is defined by the combination of FQDN, the IP address and the client serial number. FQDN is the required name attribute. It is important to notice that, a client named as Alice.cisco.com is different from the client named as Alice.cisco.com plus IPAddress 192.0.2.4.

Each enrollment transaction is uniquely associated with a transaction identifier (carried in the transactionID signed attribute (see section XXX). Because the enrollment transaction could be interrupted by various errors, including network connection errors or client reboot, the SCEP client generates a transaction identifier by calculating a hash on the public key value for which the enrollment is requested. This retains the same transaction identifier throughout the enrollment transaction, even if the client has rebooted or timed out, and issues a new enrollment request for the same key pair. It also provides the way for the CA to uniquely identify a transaction in its database. At the requester side, it generates a transaction identifier which is included in PKCSReq. If the CA returns a response of PENDING, the requester will poll by periodically sending out GetCertInitial with the same transaction identifier until either a response other than PENDING is obtained, or the configured maximum time has elapsed.

If the client times out or the client reboots, the client administrator will start another enrollment transaction with the same key pair. The second enrollment will have the transaction identifier. At the server side, instead of accepting the PKCSReq as a new enrollment request, it should respond as if another GetCertInitial message had been sent with that transaction ID. In another word, the second PKCSReq should be taken as a resynchronization message to allow the enrollment resume as the same transaction.

It is important to keep the transaction id unique since SCEP requires the same policy and same identity be applied to the same subject name and key pair binding. In the current implementation, an SCEP client can only assume one identity. At any time, only one key pair, with a given key usage, can be associated with the same identity.

The following gives several examples of client to CA transactions.

Client actions are indicated in the left column, CA actions are indicated in the right column. A blank action signifies that no message was received. Note that these examples assume that the CA

enforces the certificate-name uniqueness property defined in Section 2.1.1.1.

The first transaction, for example, would read like this:

"Client Sends PKCSReq message with transaction ID 1 to the CA. The CA signs the certificate and constructs a CertRep Message containing the signed certificate with a transaction ID 1. The client receives the message and installs the certificate locally."

```
Successful Enrollment Case: no manual authentication
PKCSReq (1)             ----------> CA Signs Cert
Client Installs Cert    <---------- CertRep (1) SIGNED CERT

Successful Enrollment Case: manual authentication required
PKCSReq (10)            ----------> Cert Request goes into Queue
Client Polls            <---------- CertRep (10) PENDING
GetCertInitial (10)     ----------> Still pending
Client Polls            <---------- CertRep (10) PENDING
GetCertInitial (10)     ----------> Still pending
Client Polls            <---------- CertRep (10) PENDING
GetCertInitial (10)     ----------> Still pending
Client Polls            <---------- CertRep (10) PENDING
GetCertInitial (10)     ----------> Cert has been signed
Client Installs Cert    <---------- CertRep (10) SIGNED CERT

Resync Case - CA Receive and Signs PKCSReq, Client Did not receive
CertRep:
PKCSReq (3)           ----------> Cert Request goes into queue
                      <---------- CertRep (3) PENDING
GetCertInitial (3)    ---------->
                      <---------- CertRep (3) PENDING
GetCertInitial (3)    ----------->
                      <----------- CA signed Cert and sent back
                                   CertRep(3)
(Time Out)
PKCSReq (3)           ----------> Cert already signed, sent back to
                                  client
Client Installs Cert  <---------- CertRep (3) SIGNED CERT
```

Case when NVRAM is lost and client has to generate a new key pair, there is no change of name information:

```
PKCSReq (4)           ----------> CA Signs Cert
Client Installs Cert  <---------- CertRep (4) SIGNED CERT
(Client looses Cert)
PKCSReq (5)           ----------> There is already a valid cert with
                                  this DN.
Client Admin Revokes  <---------- CertRep (5) OVERLAPPING CERT ERROR
PKCSReq (5)           ----------> CA Signs Cert
Client Installs Cert  <---------- CertRep (5) SIGNED CERT
Case when client admin resync the enrollment using a different PKCS#10:
PKCSReq (6)             ----------> CA Signs Cert
                        <---------- CertRep (6) SIGNED CERT

(Client timeout and admin starts another enrollment with a different
PKCS#10, but the same transaction id)

PKCSReq (6)  with different PKCS#10
                        ----------> There is already a valid cert with
                                    this entity (by checking FQDN).
                        <---------- CertRep (6) INVALID PKCS#10 CERT
                                     ERROR

Client admin either revokes the existing certificate or corrects the
error by enrolling with the same PKCS#10 as the first PKCSReq(6)

PKCSReq (6)             ----------> CA find the existing Cert
Client Installs Cert    <---------- CertRep (6) SIGNED CERT
Resync case when server is slow in response:
PKCSReq (13)    ---------->  Cert Request goes into Queue
                <----------  CertRep (13) PENDING
GetCertInitial  ----------> Still pending
                <----------  CertRep (13) PENDING
GetCertInitial  ---------->  Still pending
                <----------  CertRep (13) PENDING
GetCertInitial  ---------->  Still pending
                <----------  CertRep (13) PENDING
GetCertInitial  ---------->  Still pending
(TimeOut)       <----------  CertRep (13) PENDING
* Case 1
PKCSReq (13)          ---------->  Still pending
Client polls          <----------  CertRep (13) PENDING
GetCertInitial       ---------->  Crete has been signed
Client Installs Crete  <----------  CertRep (13) SIGNED CERT
* Case 2
PKCSReq (13)          ----------> Cert has been signed
Client Installs Cert  <---------- CertRep (13) SIGNED CERT
```

Appendix F. CA Capabilities

The response for a GetCACaps message is a list of CA capabilities, in plain text, separated by <LF> characters, as follows (quotation marks are NOT sent):

Keyword	Description
"GetNextCACert"	CA Supports the GetNextCACert message.
"POSTPKIOperation"	PKIOPeration messages may be sent via HTTP POST.
"Renewal"	Clients may use current certificate and key to authenticate an enrollment request for a new certificate.
"SHA-512"	CA Supports the SHA-512 hashing algorithm in signatures and fingerprints.
"SHA-256"	CA Supports the SHA-256 hashing algorithm in signatures and fingerprints.
"SHA-1"	CA Supports the SHA-1 hashing algorithm in signatures and fingerprints.
"DES3"	CA Supports triple-DES for encryption.

The client should use SHA-1, SHA-256, or SHA-512 in preference to MD5 hashing if it is supported by the CA.

A client MUST be able to accept and ignore any unknown keywords that might be sent back by a CA.

If none of the above capabilities are supported by the CA, no data is returned.

The appropriate HTML headers are returned in any case.

Example:

```
GET /cgi-bin/pkiclient.exe?operation=GetCACaps&message=myca
```

might return:

```
GetNextCACert POSTPKIOperation
```

This means that the CA supports the GetNextCACert message and allows PKIOperation messages (PKCSreq, GetCert, GetCertInitial...) to be sent using HTTP POST.

Appendix G. Certificate Renewal and CA Key Rollover

To renew a client certificate, use the PKCSreq message and sign it with the existing client certificate instead of a self-signed certificate.

To obtain the new CA certificate prior to the expiration of the current one, use the GetNextCACert message if the CA supports it.

To obtain a new client certificate signed by the new CA certificate, use the new CA or RA certificate in the message envelope. Example:

```
GetNextCACert            ---------->
                         <---------- CertRep (3) New CA certificate

PKCSReq* (1)             ----------> CA Signs certificate with NEW key
Client Stores Cert       <---------- CertRep (3) Certificate issued
for installation when                from NEW CA certificate and keypair
existing cert expires.
```

*enveloped for new CA or RA cert and keypair. The CA will use the envelope to determine which key and certificate to use to issue the client certificate.

Appendix H. PKIOperation via HTTP POST Message

If the remote CA supports it, any of the PKCS#7-encoded SCEP messages may be sent via HTTP POST instead of HTTP GET. This is allowed for any SCEP message except GetCACert, GetCACertChain, GetNextCACert, or GetCACaps. In this form of the message, Base 64 encoding is not used.

```
POST /cgi-bin/pkiclient.exe?operation=PKIOperation HTTP/1.0
Content-Length: <length of data>

<binary PKCS7 data>
```

The client can verify that the CA supports SCEP messages via POST by looking for the "POSTPKIOperation" capability (See Appendix F).

Authors' Addresses

Andrew Nourse
Cisco Systems, Inc.
510 McCarthy Drive
Milpitas, CA
USA

Email: nourse@cisco.com

Xiaoyi Liu
Cisco Systems, Inc.
510 McCarthy Drive
Milpitas, CA
USA

Email: xliu@cisco.com

Jan Vilhuber (editor)
Cisco Systems, Inc.
510 McCarthy Drive
Milpitas, CA
USA

Email: vilhuber@cisco.com

Cheryl Madson
Cisco Systems, Inc.
510 McCarthy Drive
Milpitas, CA
USA

Email: cmadson@cisco.com