Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 8, 2011

M. Pritikin, Ed.
A. Nourse
J. Vilhuber
Cisco Systems, Inc
March 7, 2011

Cisco Systems' Simple Certificate Enrollment Protocol
draft-nourse-scep-22

Abstract

This document specifies the Simple Certificate Enrollment Protocol, a PKI communication protocol which leverages existing technology by using PKCS#7 and PKCS#10 over HTTP. 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.

Status of this Memo

Copyright Notice

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 certificate revocation status information, although these functions are often provided by distinct protocols, e.g. CRLs [RFC4523] or OCSP [RFC2560] for X.509 certificates.

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, which 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]. Environments that do not require interoperability with SCEP implementations SHOULD use the above-mentioned, PKIX-standard certificate management protocols. Even when interoperability with the installed base of SCEP implementations is required, implementers are encouraged to support one of these comprehensive standards track certificate management protocols in addition to the protocol defined in this specification. This implementation strategy balances near-term requirements for interoperability with-longer term security goals.

As a reflection of the history of SCEP implementations some of the operations described in this document are indicated as 'SHOULD' or 'MAY' where a stricter protocol specification might have indicated a 'MUST'.

The protocol supports the following general operations:

- CA and RA public key distribution
- Certificate enrollment
- Certificate query
- CRL query

SCEP makes extensive use of PKCS#7 [RFC2315] and PKCS#10 [RFC2986].

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. SCEP Overview

This section provides a high level overview of the functionality of SCEP.

2.1. SCEP Entities

The entity types defined in SCEP are

- o the Requester (Section 2.1.1) (e.g., IPSEC clients)
- o the Server, which may be either a Certification Authority (CA) (Section 2.1.2) or a Registration Authority (RA) (Section 2.1.3)

2.1.1. Requester

The requester is sometimes called a "client" in this document. It is the client of the SCEP exchange.

The requester MAY submit SCEP messages for itself or it MAY submit SCEP messages on behalf of peers as described in Registration Authority (Section 2.1.3). This section focuses on the requester that is obtaining certificates for its own use.

Before a requester can start a PKI transaction, it MUST have at least one appropriate key pair (e.g. RSA) for use when signing the SCEP pkiMessage (Section 3.1).

The message types, being based on PKCS#7 [RFC2315] and PKCS#10 [RFC2986] fully support algorithm agility but the requester has to use a key type that is supported by the server. RSA is the only algorithm supported by current implementations.

A requester MUST have the following information locally configured:

1. The Certification Authority IP address or fully qualified domain name

2. The Certification Authority HTTP CGI script path

3. The identifying information that is used for authentication of the Certification Authority in Section 4.1.1. This information MAY be obtained from the user, or presented to the end user for manual authorization during the protocol exchange (e.g. the user indicates acceptance of a fingerprint via a user-interface element).

The requester MUST have MESSAGE information configured if the Certification Authority requires it (see Section 5.1).

The requester MAY maintain multiple independent configurations appropriate for multiple Certification Authorities. Doing so does not effect the protocol operation and is not in scope of this document.

Certificate requests for certificates whose purpose is a specific solution are encouraged to conform to the solution's profile, e.g. [RFC4945] Section 5 for IKE/IPsec certificates.

2.1.2. Certification Authority

An SCEP Certification Authority (CA) is the entity that signs client certificates. The CAs name appears in the issuer field of resulting certificates.

Before any PKI operations can occur, the SCEP CA server obtains a 'CA' certificate that matches the profile in [RFC5280]. This MAY be a CA certificate that was issued by a higher level CA.

The SCEP server CA certificate MUST be provided out-of-band to the SCEP requester. The CA certificate fingerprint MAY be used to authenticate a CA Certificate distributed by the GetCACert response (Section 4.1.1). The fingerprint is created by calculating a SHA-1, SHA-256, SHA-512, or MD5 hash on the whole CA certificate.

The certification authority MUST either include a cRLDistributionPoint extension in every certificate it issues or answer CRL queries itself, in which case it SHOULD be online at all times. The certification authority SHOULD either answer certificate queries or make certificates available via LDAP.

A certification authority may enforce any arbitrary policies, including name uniqueness policies, and apply them to certification requests. The certification authority MAY reject any request. If the client has already been issued a certificate for this keypair the server MAY return the previously created certificate. The requester

MUST NOT assume any of the fields in the certification request, except for the public key, will be the same in the certificate issued.

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. The CA SHOULD return the status of the original transaction, including the certificate if it was granted. The CA SHOULD NOT create a new transaction unless the original certificate has been revoked, or the transaction arrives more than halfway through the validity period of the original certificate.

2.1.3. Registration Authority

An SCEP Registration Authority (RA) is an SCEP server that performs validation and authorization checks of the SCEP requester but forwards the certification requests to the CA. The RAs name does not appear in the issuer field of resulting certificates.

The RA MUST return the RA certificate, in addition to the CA certificate, in the GetCACert Response (see Section 5.2.1.1.2). The existence of an RA certificate in this response indicates to the client that an RA is in use. In order to securely communicate with an RA using SCEP Secure Message Objects (Section 3) the client MUST use the RA's keys instead of the CA's keys to sign the messages.

In order to service certification requests the RA must pass the requests to the CA server for signing. The RA MAY use SCEP to communicate with the CA, in which case the RA acts as both an SCEP server (between the client and the RA) and an SCEP requester (between the RA and the CA). The RA MAY respond to client certificate requests with a PENDING response while communicating with the CA; for example if the CA must manually authorize a certification request and thus returns PENDING to the RA the RA may respond with PENDING to the client while polling the CA.

Communication between the RA and the CA MAY be over other protocols such as Certificate Management over CMS [RFC5272].

2.2. 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.

The communication between the requester and the certification authority are secured using SCEP Secure Message Objects (Section 3) which specifies how PKCS#7 [RFC2315] is used to encrypt and sign the data. In order to perform the signing operation the client uses an appropriate local certificate:

1. If the requesting system already has a certificate issued by the SCEP server, and the server supports renewal (see Appendix C), that certificate SHOULD be used.

2. If the requesting system has no certificate issued by the new CA, but has credentials from an alternate CA the certificate issued by the alternate CA MAY be used. Policy settings on the new CA will determine if the request can be accepted or not. This is useful when enrolling with a new administrative domain; by using a certificate from the old domain as credentials.

3. If the requester does not have an appropriate existing certificate, then a locally generated self-signed certificate MUST be used instead. The self-signed certificate MUST use the same subject name as in the PKCS#10 request.

During the certificate enrollment, the requester MUST use the selected certificate to sign the PKCS#7 [RFC2315] (see Section 3). The server CertResp uses this signing certificate when encrypting the response (see Section 3.2.2).

When the certification authority creates the PKCS#7 [RFC2315] envelope on the issued certificate, it SHOULD use the public key, issuer name, and serial number conveyed in the above included certificate. This will inform the end entity of 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 MUST reflect the key used to encrypt the request.

2.3. Enrollment authorization

There are two mechanisms for automated enrollment authorization.

When the client uses a certificate which is not self-signed to sign SCEP messages the server MAY use this certificate to authenticate the client and determine the appropriate authorization. In addition to the policy requirements implied by optional support of renewal, see

Appendix D, the SCEP server SHOULD implement appropriate logic to support client authentication and automated enrollment using existing client credentials that were issued by an alternate PKI hierarchy. The SCEP server MUST NOT attempt to authenticate a client based on a self-signed certificate.

PKCS#10 [RFC2986] specifies a PKCS#9 [RFC2985] challengePassword attribute to be sent as part of the enrollment request. SCEP optionally uses this challengePassword to allow for unauthenticated authorization of enrollment requests. The PKCS#7 [RFC2315] envelope protects the privacy of the challenge password.

When utilizing the challengePassword, the server distributes a shared secret to the requester which will uniquely associate the enrollment request with the requester. 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.

A client that is performing certificate renewal as per Appendix D SHOULD send an empty challenge password (i.e. use the empty string as the challenge password) but MAY send the originally distributed challenge password in the challengePassword attribute. In the former case the SCEP CA MUST authenticate the request based on the certificate used to sign the renewal request. In the latter case the SCEP CA MAY use either the challengePassword or the previously issued certificate (or both) to authenticate the request.

In the manual mode the requester's messages are placed in the PENDING state until the CA operator authorizes or rejects them. Manual authorization is used when the client has only a self-signed certificate and/or a challengePassword is not available. The SCEP server MAY either reject unauthorized certification requests or mark them for manual authorization according to server configuration.

The requester generates a SHA-1, SHA-256, SHA-512, or MD5 'fingerprint' of the PKCS#10 [RFC2986] (before PKCS#7 [RFC2315] enveloping and signing). This fingerprint is sent to the CA operator using an out-of-band method. The CA operator MUST compared this fingerprint to a locally generated fingerprint based on the message received during the SCEP exchange.

SCEP clients and CAs (or RAs, if appropriate) MUST support display of this fingerprint to the operator to enable this authorization method. The out-of-band distribution and comparison of fingerprints is not covered by this document.

2.4. CA/RA Certificate Distribution

If the CA and/or RA certificates have not previously been acquired by the requester in some other means, the requester MUST retrieve the CA/RA certificates before any PKI operation (Section 3) can be started.

Since no public key has yet been exchanged between the requester and the CA/RA, the messages cannot be secured using PKCS#7 [RFC2315], and the data is instead transferred in the clear.

If an RA is in use, a certificates-only PKCS#7 [RFC2315] SignedData with a certificate chain consisting of both RA and CA certificates is returned. Otherwise the CA certificate itself is returned. The transport protocol (Section 5) MUST indicate which one is returned.

After the requester gets the CA certificate, it MUST authenticate the CA certificate by comparing the CA certificate fingerprint (see Section 2.1.2) with the locally configured, out-of-band distributed, identifying information.

Since the optional RA certificates are signed by the CA there is no need to authenticate them against the out-of-band data. Clients MUST verify the RA certificate signature before use during protocol exchanges. Clients MUST verify the authorization of the RA certificates. The authorization mechanism is specified by the CA administrator and is out of scope for this document.

Because a long time can pass between queries from a requester to a CA/RA and because RA certificates can change at any time, it is recommended that a requester not store RA certificates. Instead, the requester SHOULD retrieve the CA/RA certificates before each operation.

2.5. Certificate Enrollment

A requester starts an enrollment (Section 3.2.1) transaction by creating a certificate request using PKCS#10 [RFC2986] and sends it to the CA/RA enveloped using the PKCS#7 (Section 3).

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 authorize the request.

If the CA/RA returns a CertRep (Section 3.2.2) message with status set to PENDING, the requester enters into polling mode by periodically sending a GetCertInitial (Section 3.2.3) PKI message to

the CA/RA, until the CA/RA operator completes the manual authentication (approving or denying the request).

In general, the requester will send a single PKCSReq (Section 3.2.1) message, followed by 0 or more GetCertInitial (Section 3.2.3) messages, if polling mode is entered.

In general, the CA/RA will send 0 or more CertRep (Section 3.2.2) messages with status set to PENDING, followed by a single CertRep (Section 3.2.2) with status set to either SUCCESS or FAILURE.

2.5.1. Client State Transitions

The requester state transitions during enrollment operation are indicated in Figure 1.

```
                         GetCertInitial
                           +----<---+
                           |        | CertRep(PENDING),
                           |        | GetCertInitial send-timeout,
                           |        | new-poll timer
                           |        |
   [CERT-NONEXISTANT] -----+---> [CERT-REQ-PENDING]      [CERT-ISSUED]
         ^             PKCSReq      |          |               ^
         |                          |          |               |
         |                          |          +---------------+
         |                          |           CertRep(SUCCESS)
         +--------------------------+
         CertRep(FAILURE),
         PKCSReq send-timeout,
         max-time/max-polls exceeded
```

Figure 1: State Transition Diagram

Certificate enrollment starts at the state CERT-NONEXISTANT.

Sending a PKCSReq message changes the state to CERT-REQ-PENDING. If there is no response, or sending is not possible, the state reverts back to CERT-NONEXISTANT.

Receiving a CertRep message with pkiStatus set to SUCCESS changes the state to CERT-ISSUED.

Receiving a CertRep message with pkiStatus set to FAILURE changes the state to CERT-NONEXISTANT.

If the server sends back a CertRep message with pkiStatus set to PENDING, the requester will keep polling by sending a GetCertInitial message to the server, until either a CertRep message with status set

to SUCCESS or FAILURE is received, or the maximum number of polls has been exceeded.

If the maximum number of polls has been exceeded or a CertRep message with pkiStatus set to FAILURE is received while in the CERT-REQ-PENDING state, the end entity will transition to the CERT-NONEXISTANT state, and the SCEP client can eventually initiate 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 certification authority can recognize this as an existing transaction instead of a new one.

A successful transaction in automatic mode:

```
         REQUESTER                           CA SERVER

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

Figure 2: Automatic mode transaction

A successful 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.
```

Figure 3: Manual mode transaction

2.6. Certificate Access

A certificate query message is defined for clients to retrieve a copy of their own certificates from the CA. It allows clients that do not store their certificate locally to obtain a copy when needed. This functionality is not intended to provide a general purpose

certificate directory service.

To query a certificate from the certification authority, a requester sends a request consisting of the certificate's issuer name and 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 (Section 3.2.4) message and one CertRep (Section 3.2.2) message, as shown in Figure 4.

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

Figure 4: GetCert Transaction

2.7. CRL Access

SCEP clients request a CRL via one of two methods:

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.

2. If the CA does not support CDP's, a CRL query is composed by creating a GetCRL message consisting of the issuer name and serial number from a certificate within the scope of the CRL to be retrieved (e.g. from a certificate to be validated).

The server SHOULD NOT support the GetCRL method because:

o it does not scale well due to the unnecessary cryptography (see, Section 8.8)

o it requires the CA to be a high-availability service

o only limited information to determine the CRL scope is provided (see [RFC5280] Section 5).

The message is sent to the SCEP server in the same way as the other SCEP requests: The transaction to retrieve a CRL consists of one GetCRL PKI message and one CertRep PKI message, which contains only the CRL (no certificates), as shown in Figure 5.

On receipt of this message, the SCEP server MAY use the IssuerAndSerial information to return an appropriate CRL.

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

Figure 5: GetCRL Transaction

2.8. Certificate Revocation

SCEP does not specify a method to request certificate revocation.

In order to revoke a certificate, the requester must contact the CA server operator using a non-SCEP defined mechanism. Although the PKCS#10 [RFC2986] challengePassword is used by SCEP for enrollment authorization (see Enrollment authorization (Section 2.3)) this does not inhibit the CA server from maintaining a record of the challengePassword to use during subsequent revocation operations as implied by [RFC2985].

3. SCEP Secure Message Objects

PKCS#7 [RFC2315] is a general enveloping mechanism that enables both signed and encrypted transmission of arbitrary data.

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

SCEP messages that require confidentiality use two layers of PKCS#7, as shown in Figure 6. By applying both enveloping and signing transformations, the SCEP message is protected both for the integrity of its end-to-end transaction 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 invoking security handling procedures specific to the information portion being processed.

Some messages do not require enveloping, in which case the EnvelopedData in Figure 6 is omitted.

```
ContentType = SignedData (called pkiMessage)
  SignerInfo
     Signature
     authenticatedAttributes
        transactionID
        messageType
        pkiStatus
        failInfo
        senderNonce
        recipientNonce
        etc
  ContentInfo type = EnvelopedData (called pkcsPKIEnvelope; optional)
     RecipientInfo
     ContentInfo type = Data
        messageData
```

Figure 6: PKCS#7 Layering

Description:

- o The outer PKCS#7 is a pkiMessage (Section 3.1).

- o The SignedData ContentInfo, if present (e.g. FAILURE and PENDING CertRep messages will lack any signed content), MUST be a pkcsPKIEnvelope (Section 3.1.2).

When a particular SCEP message carries data, this data is carried in the messageData.

Note: The remainder of this document will refer only to 'messageData', but it is understood to always be encapsulated in the pkcsPKIEnvelope (Section 3.1.2). The format of the data in the messageData is defined by the messageType attribute (see Section 3.1.1) of the SignedData. If there is no messageData to be transmitted, the entire pkcsPKIEnvelope MUST be omitted.

3.1. SCEP pkiMessage

The basic building block of all secured SCEP messages is the SCEP pkiMessage. It consists of an PKCS#7 signed-data content type, as defined in PKCS#7 [RFC2315] Section 9. The following restrictions apply:

- o version MUST be 1

- o the contentType in contentInfo MUST be data ({pkcs-7 1}) as defined in PKCS#7 [RFC2315] Section 8.

- o The signed content, if present (e.g. FAILURE and PENDING CertRep messages will lack any signed content), MUST be a pkcsPKIEnvelope (Section 3.1.2), and must match the messageType attribute.

- o The SignerInfo MUST contain a set of authenticatedAttributes (see PKCS#7 [RFC2315] Section 9.2 as well as Section 3.1.1 in this document). All messages MUST contain

  * an SCEP transactionID attribute

  * an SCEP messageType attribute

  * an SCEP senderNonce attribute

  * any attributes required by PKCS#7 [RFC2315] Section 9.2

  If the message is a response, it MUST also include

  * an SCEP pkiStatus attribute

  * an SCEP recipientNonce attribute

3.1.1. Signed Transaction Attributes

The following transaction attributes are encoded as authenticated attributes, and are carried, as specified in PKCS#7 [RFC2315] Section 9.2, in the SignerInfo for this signedData.

Please refer to Appendix A for the OID definitions.

Attribute	Encoding	Comment
transactionID	PrintableString	Hash 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	OCTET STRING	
recipientNonce	OCTET STRING	

Transaction Attributes

The attributes are detailed in the following sections.

3.1.1.1. transactionID

A PKI operation is a transaction consisting of the messages exchanged between a requester and the server. The transaction identifier is a string generated by the client when starting a transaction. The client MUST generate a unique string as the transaction identifier, which MUST be used for all PKI messages exchanged for a given enrollment, encoded as a PrintableString.

The transactionID SHOULD be 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 automatically generate the same transactionID for any given keypair. The SCEP protocol requires that transactionIDs be unique, so that subsequent polling queries can be matched with previous transactions. When separate signing and encryption certificates are requested by the client, using distinct keypairs ensures that distinct transactionIDs are also used.

When using the certificate query and CRL query messages defined in this protocol, the transactionID is required so that the requester can match the response message with the outstanding request message. When using LDAP to query the certificate and the CRL, the behavior is specified by the LDAP protocol. For a non-enrollment message (for example GetCert and GetCRL), the transactionID SHOULD be a number unique to the client.

3.1.1.2. messageType

The messageType attribute specifies the type of operation performed by the transaction. This attribute MUST be included in all PKI messages. Currently, the following message types are defined:

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

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

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

- GetCert (21) -- Retrieve a certificate

- GetCRL (22) -- Retrieve a CRL

3.1.1.3. pkiStatus

All response messages MUST include transaction status information, which is defined as pkiStatus attribute:

- o SUCCESS (0) -- request granted

- o FAILURE (2) -- request rejected. When pkiStatus is FAILURE, the failInfo attribute, as defined in Section 3.1.1.4, MUST also be present.

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

3.1.1.4. failInfo

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

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

- o badMessageCheck (1) -- integrity check failed

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

- o badTime (3) -- The signingTime attribute from the PKCS#7 authenticatedAttributes was not sufficiently close to the system time (see Section 3.1.1.6).

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

3.1.1.5. senderNonce and recipientNonce

The attributes of senderNonce and recipientNonce are a 16 byte random number generated for each transaction. These are intended to prevent replay attacks.

When a requester sends a PKI message to the server, a senderNonce MUST be included in the message.

The recipient SHOULD copy the senderNonce into the recipientNonce of the reply as a proof of liveliness.

The requester SHOULD verify that the recipientNonce of the reply matches the senderNonce it sent in the request.

3.1.1.6. signingTime Attribute

The signingTime attribute is defined in [RFC2985] Section 5.3.3, and is carried as specified in [RFC2315] Section 9.2. This attribute is optional.

3.1.2. SCEP pkcsPKIEnvelope

The information portion of a SCEP message is carried inside an enveloped-data content type, as defined in PKCS#7 [RFC2315] Section 10, with the following restrictions:

- o version MUST be 0
- o contentType in encryptedContentInfo MUST be data ({pkcs-7 1}) as defined in PKCS#7 [RFC2315] Section 8.
- o encryptedContent MUST be the SCEP message being transported (see Section 4), and must match the messageType authenticated Attribute in the pkiMessage.

The PKCS#7 [RFC2315] content-encryption key (see Section 10, step 2) is encrypted using the public key of the recipient of the message, i.e. the RA or the CA public key (if sent from the requester), or the requester public key (if sent as a reply to the requester).

3.2. SCEP pkiMessage types

All of the messages in this section are pkiMessages (Section 3.1), where the type of the message MUST be specified in the 'messageType' authenticated Attribute. Each section defines a valid message type, the corresponding messageData formats, and mandatory authenticated attributes for that type.

3.2.1. PKCSReq

The messageData for this type consists of a DER-encoded PKCS#10 Certification Request [RFC2986]. The certification request MAY contain any fields defined in PKCS#10 [RFC2986], and 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.

In addition to the authenticatedAttributes required for a valid PKCS#7 [RFC2315], this pkiMessage MUST include the following attributes:

- a transactionID (Section 3.1.1.1) attribute
- a messageType (Section 3.1.1.2) attribute set to PKCSReq
- a senderNonce (Section 3.1.1.5) attribute

The pkcsPKIEnvelope for this message type is protected using the public key of the recipient as detailed in Section 3.1.2, e.g. either the CA or RA public key.

3.2.2. CertRep

The messageData for this type consists of a DER-encoded degenerate certificates-only Signed-data (Section 3.3). The exact content required for they reply depends on the type of request this message is a reply to. They are detailed in Table 1 and in Section 4.

In addition to the authenticatedAttributes required for a valid PKCS#7 [RFC2315], this pkiMessage MUST include the following attributes:

- the transactionID (Section 3.1.1.1) attribute copied from the request we are responding to
- a messageType (Section 3.1.1.2) attribute set to CertRep
- a senderNonce (Section 3.1.1.5) attribute
- a recipientNonce attribute (Section 3.1.1.5) copied from the senderNonce from the request we are responding to.
- a pkiStatus (Section 3.1.1.3) set to the status of the reply.

The pkcsPKIEnvelope for this message type is protected using the public key of the recipient as detailed in Section 3.1.2. For example if a self-signed certificate was used to send the original request then this self-signed certificate's public key is used to encrypt the content-encryption key of the SUCCESS response's pkcsPKIEnvelope.

3.2.2.1. CertRep SUCCESS

When the pkiStatus attribute is set to SUCCESS, the messageData for this message consists of a DER-encoded degenerate certificates-only Signed-data (Section 3.3). The content of this degenerate certificates-only Signed-Data depends on what the original request was, as outlined in Table 1.

Request-type	Reply-contents
PKCSReq	the reply MUST contain at least the issued certificate in the certificates field of the Signed-Data. The reply MAY contain additional certificates, but the issued certificate MUST be the first in the list. The reply MUST NOT contain a CRL. All returned certificates MUST conform to [RFC5280].
GetCertInitial	same as PKCSReq
GetCert	the reply MUST contain at least the requested certificate in the certificates field of the Signed-Data. The reply MAY contain additional certificates, but the requested certificate MUST be the first in the list. The reply MUST NOT contain a CRL. All returned certificates MUST conform to [RFC5280].
GetCRL	the reply MUST contain the CRL in the crls field of the Signed-Data. The reply MUST NOT contain a certificate. The CRL MUST be a valid CRL according to [RFC5280].

Table 1: CertRep Types

3.2.2.2. CertRep FAILURE

When the pkiStatus attribute is set to FAILURE, the reply MUST also contain a failInfo (Section 3.1.1.4) attribute set to the appropriate error condition describing the failure. The pkcsPKIEnvelope (Section 3.1.2) MUST be omitted.

3.2.2.3. CertRep PENDING

When the pkiStatus attribute is set to PENDING, the pkcsPKIEnvelope (Section 3.1.2) MUST be omitted.

3.2.3. GetCertInitial

The messageData for this type consists of a DER-encoded IssuerAndSubject (Section 3.2.3.1). The issuer is set to the issuerName from the certification authority from which we are issued certificates. The Subject is set to the SubjectName we used when requesting the certificate.

In addition to the authenticatedAttributes required for a valid PKCS#7 [RFC2315], this pkiMessage MUST include the following

attributes:

- the same transactionID (Section 3.1.1.1) attribute from original PKCSReq message

- a messageType (Section 3.1.1.2) attribute set to GetCertInitial

- a senderNonce (Section 3.1.1.5) attribute

3.2.3.1. IssuerAndSubject

Similar to the IssuerAndSerial defined in PKCS#7 [RFC2315] Section 6.7, we need to define an IssuerAndSubject ASN.1 type (Figure 7).

The ASN.1 definition of the issuerAndSubject type is as follows:

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

Figure 7: IssuerAndSubject ASN.1

3.2.4. GetCert

The messageData for this type consists of a DER-encoded IssuerAndSerial as defined in PKCS#7 [RFC2315] Section 6.7 containing the "distinguished name of the certificate issuer and an issuer-specific certificate serial number" which uniquely identifies the certificate being requested.

In addition to the authenticatedAttributes required for a valid PKCS#7 [RFC2315], this pkiMessage MUST include the following attributes:

- a transactionID (Section 3.1.1.1) attribute

- a messageType (Section 3.1.1.2) attribute set to GetCert

- a senderNonce (Section 3.1.1.5) attribute

A self-signed certificate MAY be used in the signed envelope. This enables the requester to request their own certificate if they were unable to store it previously.

3.2.5. GetCRL

The messageData for this type consists of a DER-encoded IssuerAndSerial as defined in PKCS#7 [RFC2315] Section 6.7 along with

the issuer name and serial number from the certificate to be validated.

In addition to the authenticatedAttributes required for a valid PKCS#7 [RFC2315], this pkiMessage MUST include the following attributes:

o a transactionID (Section 3.1.1.1) attribute

o a messageType (Section 3.1.1.2) attribute set to GetCRL

o a senderNonce (Section 3.1.1.5) attribute

3.3. Degenerate certificates-only PKCS#7 Signed-data

[RFC2315] Section 9 includes a degenerate case of the PKCS#7 Signed-data content type, in which there are no signers. The use of such a degenerate case is to disseminate certificates and certificate-revocation lists. For SCEP the content field of the ContentInfo value of a degenerate certificates-only Signed-Data MUST be omitted.

When carrying certificates, the certificates are included in the 'certificates' field of the Signed-Data. When carrying a CRL, the CRL will be included in the 'crls' field of the Signed-Data.

4. SCEP Transactions

This section describes the SCEP Transactions, without explaining the transport. The transport of each message is discussed in Section 5. Some of the transaction-requests have no data to send, i.e. the only data is the message-type itself (e.g. a GetCACert message has no additional data). The use of such messages will become clearer in Section 5.

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

The order of the transactions in this section is mirrored in Section 5.2 for better organization and readability.

4.1. Get CA Certificate

To get the CA certificate(s), the requester sends a GetCACert message to the server. There is no request data associated with this message (see Section 5.2.1).

4.1.1. Get CA Certificate Response Message Format

The response depends on whether the responding server has RA certificates or only a single CA certificate. The server MUST indicate which response it is sending via the transport protocol used (see Section 5.2.1).

All returned certificates MUST conform to [RFC5280].

A fingerprint is generated using the SHA1, SHA256, SHA512 or MD5 hash algorithm on the whole CA certificate received by the requester (regardless of the presence of RA certificates). If the requester does not have a certificate path to a trust anchor certificate, this fingerprint may be used to verify the certificate, by some positive out-of-band means, such as a phone call or prior configuration.

4.1.1.1. CA Certificate Response Message Format

If the server is a certification authority and does not have any RA Certificates, the response consists of a single DER-encoded X.509 CA certificate.

4.1.1.2. CA/RA Certificate Response Message Format

If the server has RA Certificates, the response consists of a DER-encoded degenerate certificates-only Signed-data (Section 3.3) containing the CA certificate and RA certificates.

4.2. Certificate Enrollment

A PKCSReq (Section 3.2.1) message is used to perform a certificate enrollment transaction.

The reply MUST be a CertRep (Section 3.2.2) message sent back from the server, indicating SUCCESS, FAILURE, or PENDING.

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

Postcondition: The requester recieves the certificate, the request is rejected, or the request is pending. A pending response might indicate that manual authentication is necessary.

4.2.1. Certificate Enrollment Response Message

If the request is granted, a CertRep (Section 3.2.2) message with pkiStatus set to SUCCESS is returned. The reply MUST also contain

the certificate (and MAY contain any other certificates needed by the requester). The issued certificate MUST be the first in the list.

If the request is rejected, a CertRep (Section 3.2.2) message with pkiStatus set to FAILURE is returned. The reply MUST also contain a failInfo attribute.

If the the CA is configured to manually authenticate the requester, a CertRep (Section 3.2.2) message with pkiStatus set to PENDING MAY be returned. The CA MAY return a PENDING for other reasons.

4.3. Poll for Requester Initial Certificate

Triggered by a CertRep (Section 3.2.2) with pkiStatus set to PENDING, a requester will enter the polling state by periodically sending GetCertInitial (Section 3.2.3) 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 (or maximum number of polls) is exceeded.

Since GetCertInitial is part of the enrollment, the messages exchanged during the polling period MUST carry the same transactionID attribute as the previous PKCSReq. A server receiving a GetCertInitial for which it does not have a matching PKCSReq MUST ignore this request.

Since at this time the certificate has not been issued, the requester can only use its own subject name (which was contained in the original PKCS#10 sent via PKCSReq) to identify the polled certificate request. Since there can be multiple outstanding requests from one requester (for example, if different keys and different key-usages were used to request multiple certificates), the transaction ID must also be included, to disambiguate between multiple requests.

PreCondition: The requester has received a CertRep with pkiStatus set to PENDING.

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.

4.3.1. Polling Response Message Format

The response messages for GetCertInitial are the same as in Section 4.2.1.

4.4. Certificate Access

A requester can query an issued certificate from the SCEP server, as long as the requester knows the issuer name and the issuer assigned certificate serial number. This transaction is not intended to provide a generic certificate directory service.

This transaction consists of one GetCert (Section 3.2.4) message sent to the server by a requester, and one CertRep (Section 3.2.2) message sent back from the server.

PreCondition: The certification authority has issued the queried certificate and the issuer assigned serial number is known.

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

4.4.1. Certificate Access Response Message Format

In this case, the CertRep from the server is same as in Section 4.2.1, except that the server will only either grant the request (SUCCESS) or reject the request (FAILURE).

4.5. CRL Access

Clients MAY request a CRL from the SCEP server as described in Section 2.7.

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

PostCondition: CRL sent back to the requester.

4.5.1. CRL Access Response Message Format

The CRL is sent back to the requester in a CertRep (Section 3.2.2) message. The information portion of this message is a degenerate certificates-only Signed-data (Section 3.3) that contains only the most recent CRL in the crls field of the Signed-Data.

The server MAY return any appropriate CRL.

4.6. Get Next Certification Authority Certificate

When a CA certificate is about to expire, clients need to retrieve the CA's next CA certificate (i.e. the Rollover Certificate). This is done via the GetNextCACert message. There is no request data associated with this message (see Section 5.2.6).

4.6.1. Get Next CA Response Message Format

The response consists of a SignedData PKCS#7 [RFC2315], signed by the current CA (or RA) signing key.

Clients MUST validate the signature on the the SignedData PKCS#7 [RFC2315] before accepting any of its contents.

The content of the SignedData PKCS#7 [RFC2315] is a degenerate certificates-only Signed-data (Section 3.3) message containing the new CA certificate and any new RA certificates, as defined in Section 5.2.1.1.2, to be used when the current CA certificate expires.

If the CA (or RA) does not have the Rollover certificate(s) it MUST reject the request. It SHOULD also remove the GetNextCACert setting from the capabilities until it does have rollover certificates.

If there are any RA certificates in this response, clients MUST check that these RA certificates are signed by the CA, and MUST check authorization of these RA certificates (see Section 2.1.3).

5. SCEP Transport

HTTP is used as the transport protocol for SCEP Message Objects.

5.1. HTTP "GET" Message Format

SCEP uses the HTTP "GET" messages to request information from the CA. The following defines the syntax of a HTTP GET message sent from a requester to a certification authority server:
"GET" CGI-PATH CGI-PROG "?operation=" OPERATION "&message=" MESSAGE

where:

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

- 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.

- OPERATION depends on the SCEP transaction and is defined in the following sections.

- o MESSAGE depends on the SCEP transaction and is defined in the following sections.

If the CA supports it, requests SHOULD be done via an HTTP POST. This is described in Appendix F.

5.1.1. Response Message Format

For each GET operation, the CA/RA server MUST return a Content-Type and appropriate response data, if any.

5.2. SCEP HTTP Messages

This section describes the OPERATION and MESSAGE values for SCEP exchanges.

5.2.1. GetCACert

The OPERATION MUST be set to "GetCACert".

The MESSAGE MAY be omitted, or it MAY be a string that represents the certification authority issuer identifier. A CA Administrator defined string allows for multiple CAs supported by one SCEP server.

5.2.1.1. GetCACert Response

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

5.2.1.1.1. CA Certificate Only Response

The response will have a Content-Type of "application/ x-x509-ca-cert".

The body of this response consists of a DER-encoded X.509 CA certificate, as defined in Section 4.1.1.1.
"Content-Type:application/x-x509-ca-cert\n\n"<DER-encoded X.509>

5.2.1.1.2. CA and RA Certificates Response

The response will have a Content-Type of "application/ x-x509-ca-ra-cert".

The body of this response consists of a DER-encoded degenerate certificates-only Signed-data (Section 3.3) containing both CA and RA certificates, as defined in Section 4.1.1.2.

```
"Content-Type:application/x-x509-ca-ra-cert\n\n"<DER-encoded PKCS7>
```

5.2.2. PKCSReq

The OPERATION MUST be set to "PKIOperation".

The MESSAGE consists of a base64-encoded DER-encoded PKCSReq SCEP message.

An example PKIOperation request might look as follows:

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

5.2.2.1. PKCSReq Response

The response will have a Content-Type of "application/x-pki-message".

The body of this response consists of a DER-encoded CertRep SCEP message defined in Section 4.2.1. The following is an example of the response:

```
"Content-Type:application/x-pki-message\n\n"<DER-encoded CertRep msg>
```

5.2.3. GetCertInitial

The OPERATION MUST be set to "PKIOperation".

The MESSAGE consists of a base64-encoded DER-encoded GetCertInitial SCEP message.

5.2.3.1. GetCertInitial Response

The body of this response consists of a DER-encoded CertRep SCEP message defined in Section 4.3.1.

5.2.4. GetCert

The OPERATION MUST be set to "PKIOperation".

The MESSAGE consists of a base64-encoded DER-encoded GetCert SCEP message.

5.2.4.1. GetCert Response

The body of this response consists of a DER-encoded CertRep SCEP message defined in Section 4.4.1.

5.2.5. GetCRL

The OPERATION MUST be set to "PKIOperation".

The MESSAGE consists of a base64-encoded DER-encoded GetCRL SCEP message.

5.2.5.1. GetCRL Response

The body of this response consists of a DER-encoded CertRep SCEP message defined in Section 4.5.1.

5.2.6. GetNextCACert

The OPERATION MUST be set to "GetNextCACert".

The MESSAGE MAY be ommitted, or it MAY be a string that represents the certification authority issuer identifier, if such has been set by the CA Administrator.

5.2.6.1. GetNextCACert Response

The response will have a Content-Type of "application/ x-x509-next-ca-cert".

The body of this response consists of a SignedData PKCS#7 [RFC2315], as defined in Section 4.6.1. (This is similar to the GetCert response but does not include any of the attributes defined in Section 3.1.1.)

```
"Content-Type:application/x-x509-next-ca-cert\n\n"
<BER-encoded SignedData<DER-encoded degenerate PKCS7>>
```

6. Contributors/Acknowledgements

The editor would like to thank all the previous editors, authors and contributors: Cheryl Madson, Xiaoyi Liu, David McGrew, David Cooper, Andy Nourse etc for their work maintaining the draft over the years. Numerous other people have contributed during the long life cycle of the draft and all deserve thanks.

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 early versions of this protocol and this document.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

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. The adversary is computationally limited, but that can manipulate data during transmission (that is, can act as a man-in-the-middle). The precise meaning of 'computationally limited' depends on the implementer's choice of one-way hash functions and cryptographic algorithms. The mandatory to implement 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 C).

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 2.2. The third goal is met through the use of a challenge password for revocation, which is chosen by the SCEP client and communicated to the CA protected by the PKCS#7 [RFC2315] encryptedData, as specified in Section 2.8.

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 issued certificates. Subsequent protocols can use the certificate in ways that either expose the identity information, or protect it, depending on the security requirements of those protocols. The motivation for the third security goal is to protect the SCEP clients from denial of service attacks.

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 key pair is generally meant for (and is usually flagged as) certificate signing ("keyCertSign") exclusively, rather than data signing ("digitalSignature") or data encryption ("dataEncipherment"). The SCEP protocol, however, uses the CA private key to both encrypt and sign PKCS#7 [RFC2315] transport messages. This is generally considered undesirable, as it widens the possibility of an implementation weakness, and provides

o another place that the private key must be used (and hence is slightly more vulnerable to exposure),

o another place where a side channel attack (say, timing or power analysis) might be used,

o another place that the attacker might somehow insert his own text, and get it signed by the private key.

While the CA key pair 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 [RFC2315] 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 key pair. 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 and encrypted by way of being encapsulated in a pkiMessage. 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 recommended 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 SHOULD 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. Nonces and Replay

In order to detect replay attacks, both sides need to maintain state information sufficient to detect an unexpected nonce value.

Since existing implementations do not copy the senderNonce from a CertRep into subsequent GetCertinitial requests, the server will never see its own nonce reflected back to it. The transactionID can be used to link together the GetCertInitial and PKCSReq messages.

8.6. Key Usage Issues

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]. If key usage is not present, the public key is assumed to be a general purpose key that may be used for all purposes.

When building a pkiMessage, clients MUST have a certificate to sign the PKCS#7 [RFC2315] signed-data (because PKCS#7 [RFC2315] requires it). Clients MUST either use an existing certificate, or create a self-signed certificate (see Section 2.3). If the certificate has a key usage extension in it, then both the SCEP client and SCEP server MUST ignore the key usage for the duration of the transaction (the key will be used for signing during the creation of the PKCSReq message, and for encryption during the creation of the CertRep message).

8.7. GetCACaps Issues

The GetCACaps response is not signed. This allows an attacker to use downgrade attacks (as well as "upgrade attacks") on the cryptographic capabilities of the CA.

8.8. 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. The information requested is public and thus signing the requests is of questionable value but also CRLs and Certificates, i.e. the respective responses, are already signed by the CA and can be verified by the recipient without requiring additional signing and encryption.

This may affect performance and scalability of the CA and could be used as an attack vector on the CA (though not an anonymous one). The use of CDPs is recommended for CRL access, as well as other ways of retrieving certificates (LDAP, direct HTTP access, etc.).

8.9. GetNextCACert

Servers implementing early versions of the SCEP draft might return an unsigned GetNextCACert response by erroneously mirroring the (unsigned) functionality of GetCACert. Client receiving such responses MUST ignored them.

GetNextCACert depends on a 'flag moment' at which every client in the PKI infrastructure switches from the current CA certificate (and client certificate) to the new CA certificate and client certificates. Proper monitoring of the network infrastructure can ensure that this will proceed as expected but any errors in processing or implementation can result in a failure of the PKI infrastructure.

9. 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.

[RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, November 2000.

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

November 2000.

[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.

[RFC4945] Korver, B., "The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX", RFC 4945, August 2007.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.

[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. 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 B. 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.

Each enrollment transaction is uniquely associated with a transaction identifier (carried in the transactionID signed attribute (see Section 3.1.1.1). 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. The requester generates the 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. This mechanism retains the same transaction identifier throughout the enrollment transaction.

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 same transaction identifier. At the server side, instead of accepting the PKCSReq as a new enrollment request, it can respond as if another GetCertInitial message had been sent with that transaction ID. The second PKCSReq should be taken as a resynchronization message to allow the enrollment to resume as the same transaction.

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.

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
                           <---------- CertRep (10) SIGNED CERT
   Client Installs Cert

   Resync Case 1 - CA Receives PKCSReq, sends PENDING, eventually grants
   the certificate and returns SUCCESS, with the certificate.  The
   SUCCESS gets lost:
   PKCSReq (3)           ----------> Cert Request goes into queue
                         <---------- CertRep (3) PENDING
   GetCertInitial (3)    ----------> Still pending
                         <---------- CertRep (3) PENDING
   GetCertInitial (3)    -----------> Cert has been signed
                           X-------- CertRep(3) SIGNED CERT
   (Time Out)
   PKCSReq (3)           ----------> Cert already granted
                         <---------- CertRep (3) SIGNED CERT
   Client Installs Cert

   Resync Case 2 - CA Receives PKCSReq, sends PENDING, PENDING reply
   gets lost:
   PKCSReq (3)           ----------> Cert Request goes into queue
                           X-------- CertRep (3) PENDING
   (Time Out)
   PKCSReq (3)           ---------->
                         <---------- CertRep (3) PENDING
   etc...
```

```
Case when the Certificate is lost, the CA arbitrarily refuses to sign
a replacement (enforcing name-uniqueness) until the original
certificate has been revoked (there is no change of name
information):
PKCSReq (4)           ----------> CA Signs Cert
                      <---------- CertRep (4) SIGNED CERT
Client Installs Cert
(Client looses Cert)
PKCSReq (5)           ----------> There is already a valid cert with
                                  this DN.
                      <---------- CertRep (5) BAD REQUEST
                                  Admin Revokes
PKCSReq (5)           ----------> CA Signs Cert
                      <---------- CertRep (5) SIGNED CERT
Client Installs Cert
```

Appendix C. CA Capabilities

C.1. 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 C.2. CA servers SHOULD support the GetCACaps message and MUST support it when they support certificate renewal using the method described in Appendix D.

C.2. CA Capabilities Response Format

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.
"SHA-256"	CA Supports the SHA-256 hashing algorithm.
"SHA-1"	CA Supports the SHA-1 hashing algorithm.

"DES3"	CA Supports the triple-DES encryption algorithm.

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

The server MUST use the texual case specified here, but clients SHOULD ignore the textual case when processing this message. A client MUST be able to accept and ignore any unknown keywords that might be sent back by a CA.

If the CA supports none of the above capabilities the SCEP server SHOULD return an empty message. A server MAY simply return an HTTP Error. A client that receives an empty message or an HTTP error SHOULD interpret the response as if none of the requested capabilities are supported by the CA.

The Content-type of the reply SHOULD be "text/plain". Clients SHOULD ignore the Content-type, as older server implementations of SCEP may send various Content-types.

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

might return:
```
GetNextCACert<LF>POSTPKIOperation
```

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

Appendix D. Client Certificate Renewal

An enrollment request that occurs more than halfway through the validity period of an existing certificate for the same subject name and key usage MAY be interpreted as a re-enrollment or renewal request and be accepted. A new certificate with new validity dates can be issued, even though the old one is still valid, if the CA policy permits. The server MAY automatically revoke the old client certificate. Clients MUST use GetCACaps (see Appendix C) to determine if the CA supports renewal. Clients MUST support servers that do not implement renewal, or that reject renewal requests.

To renew a client certificate, the client uses the PKCSreq message and signs it with the existing client certificate. The client SHOULD use a new keypair when requesting a new certificate. The client MAY

request a new certicate using the old keypair.

Appendix E. CA Key Rollover

When the CA certificate expires all certificates that have been signed by it are no longer valid. CA key rollover provides a mechanism by which the server MAY distribute a new CA certificate which is valid in the future; when the current certificate has expired. Clients MUST use GetCACaps (see Appendix C) to determine if the CA supports GetNextCACert.

To obtain the new CA certificate prior to the expiration of the current one, the client uses the GetNextCACert message.

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

Clients MUST store the not-yet-valid CA certificate, and any not-yet-valid client certificates obitained, until such time that they are valid. At which point clients switch over to using the newly valid certificates.

Example:

```
GetNextCACert            ---------->
                         <---------- New CA certificate

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

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

Appendix F. PKIOperation via HTTP POST Message

If the remote CA supports it, any of the PKCS#7 [RFC2315]-encoded SCEP messages may be sent via HTTP POST instead of HTTP GET. This is allowed for any SCEP message except GetCACert, 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 PKCS#7 data>
```

General POST Syntax

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

Authors' Addresses

Max Pritikin (editor)
Cisco Systems, Inc

Email: pritikin@cisco.com

Andrew Nourse
Cisco Systems, Inc

Email: nourse@cisco.com

Jan Vilhuber
Cisco Systems, Inc

Email: vilhuber@cisco.com