Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 16, 2016

P. Gutmann
University of Auckland
M. Pritikin
Cisco

March 15, 2016

Simple Certificate Enrolment Protocol
draft-gutmann-scep-02.txt

Abstract

This document specifies the Simple Certificate Enrolment Protocol (SCEP), a Public Key Infrastructure (PKI) communication protocol which leverages existing technology by using CMS (formerly known as PKCS #7) and PKCS #10 over HTTP. SCEP is the evolution of the enrolment protocol sponsored by Cisco Systems, which now enjoys wide support in both client and server implementations, as well as being relied upon by numerous other industry standards that work with certificates.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 16, 2016.

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 TLS [14], S/MIME [13], and and IKE/IPsec [12]. 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 update from a Certification Authority (CA).

This specification defines a protocol, Simple Certificate Enrolment 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. certificate revocation transactions, which can significantly enhance the security achieved in a PKI. The IETF protocol suite currently includes two further certificate management protocols with more comprehensive functionality: Certificate Management Protocol (CMP) [10] and Certificate Management over CMS (CMC) [9]. Environments that do not require interoperability with SCEP implementations MAY consider using the above-mentioned certificate management protocols, however anyone considering this step should be aware that the high level of complexity of these two protocols has resulted in serious interoperability problems and corresponding lack of industry support. SCEP's simplicity, while being a drawback in terms of its limited functionality, also makes deployment relatively straightforward, so that it enjoys widespread industry support and ready interoperability across a wide range of platforms. While implementers are encouraged to investigate one of the more comprehensive alternative certificate management protocols in addition to the protocol defined in this specification, anyone wishing to deploy them should proceed with caution, and consider support and interoperability issues before committing to their use.

The protocol supports the following general operations:

- CA public key distribution.
- Certificate enrolment.
- Certificate renewal/update.
- Certificate query.
- CRL query.

SCEP makes extensive use of CMS [3] and PKCS #10 [6].

1.1. Conventions Used in This Document

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 [1].

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

- The Requester, or client (Section 2.1.1).
- The Server, a Certification Authority (CA) (Section 2.1.2).

2.1.1. Requester

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

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

The message types, being based on CMS [3] and PKCS #10 [6], fully support algorithm agility but the requester has to use a key type that is supported by the server. Specifically, they must employ a PKC algorithm capable of both encryption and signing. RSA is the only widely-used algorithm that has these properties.

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 (this usually has a default value, see Section 5.1).
3. The identifying information that is used for authentication of the Certification Authority in Section 4.1.1, typically a certificate fingerprint. 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 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.

2.1.2. Certification Authority

A SCEP Certification Authority (CA) is the entity that signs client certificates. A certification authority MAY enforce any arbitrary 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.

The certification authority MAY include a cRLDistributionPoint extension in every certificate it issues, make CRLs available via HTTP [11] or LDAP, or answer CRL queries itself. In the latter case it SHOULD be online at all times.

Since the client is expected to perform encryption and signature verification using the CA certificate, the keyUsage extension in the CA certificate MUST indicate that it is valid for digitalSignature and keyEncipherment use alongside the usual CA usages of keyCertSign and/or cRLSign.

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 transactionID. The CA SHOULD return the status of the original transaction, including the certificate if it was granted.

2.1.3. CA Certificate Distribution

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

Since no public key has yet been exchanged between the requester and the CA, the messages cannot be secured using CMS [3], and the data is instead transferred in the clear.

If an intermediate CA is in use, a certificates-only CMS [3] Signed-Data message with a certificate chain consisting of all CA certificates is returned. Otherwise the CA certificate itself is returned. The transport protocol (Section 5) MUST indicate which one is returned.

The SCEP server CA certificate MAY be provided out-of-band to the SCEP requester. Alternatively, the CA certificate fingerprint MAY be used to authenticate a CA Certificate distributed by the GetCACert response (Section 4.1) or via HTTP [11]. The fingerprint is created by calculating a SHA-1, SHA-256, or SHA-512 hash over the whole CA certificate.

After the requester gets the CA certificate, it SHOULD authenticate the certificate by comparing its fingerprint with the locally configured, out-of- band distributed, identifying information. Intermediate CA certificates, if any, are signed by a higher-level CA so there is no need to authenticate them against the out-of-band data. Clients SHOULD verify intermediate-level CA certificate signatures using the issuing CA's certificate before use during protocol exchanges.

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

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 (MITM) 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 CMS [3] 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 requester does not have an appropriate existing certificate then a locally generated self-signed certificate MUST be used. The self-signed certificate SHOULD use the same subject name as in the PKCS #10 request. In this case the messageType is PKCS10Req (see Section 3.1.1.2).
2. If the requesting system already has a certificate issued by the SCEP server, and the server supports renewal (see Section 2.4), that certificate SHOULD be used. In this case the messageType is RenewalReq (see Section 3.1.1.2).

3. 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 using a certificate from the old domain as credentials. In this case the messageType is UpdateReq (see Section 3.1.1.2).

Note that although the above text describes three different types of operations, in practice most implementations always apply the first one even if an existing certificate already exists. For this reason support for the first case is mandatory while support for the latter two are optional (see Section 2.8).

During the certificate enrolment process, the requester MUST use the selected certificate's key when signing the CMS [3] envelope (see Section 3). The server's CertResp then uses the same certificate's public key when encrypting the response (see Section 3.2.2).

> [Question: This is another area where the semantics were never defined, what happens during a renewal or update? For an enrolment the signing cert contains the key that's also in the request, but what about for a renewal or update where they're quite probably different keys? Should the envelope be encrypted to the key in the request or the signing key?].

When the certification authority creates the CMS [3] envelope containing the issued certificate, it SHOULD use the public key and identifying information conveyed in the above included certificate. This will inform the end entity of which private key is needed to open the envelope.

2.3. Enrolment authorization

PKCS #10 [6] specifies a PKCS #9 [5] challengePassword attribute to be sent as part of the enrolment request. When utilizing the challengePassword, the server distributes a shared secret to the requester which will uniquely associate the enrolment request with the requester.

Inclusion of the challengePassword by the SCEP client is OPTIONAL and allows for unauthenticated authorization of enrolment requests (which, however, requires manual approval of each certificate issue, see below), or for renewal or update requests which are authenticated by being signed with an existing certificate. The CMS [3] envelope protects the privacy of the challengePassword.

A client that is performing certificate renewal or update as per Section 2.4 SHOULD omit the challengePassword but MAY send the originally distributed 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 or update request. In the latter case the SCEP CA MAY use either the challengePassword or the previously issued certificate (or both), depending on CA policy, to authenticate the request. The SCEP server MUST NOT attempt to authenticate a client based on a self-signed certificate unless it has been verified through out-of-band means such as a certificate fingerprint.

To perform the authorization in 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 that hasn't been previously authenticated by the CA and/or a challengePassword is not available. The SCEP server MAY either reject unauthorized certification requests or mark them for manual authorization according to CA policy.

2.4. Certificate Enrolment/Renewal/Update

A requester starts an enrolment (Section 3.2.1) transaction by creating a certificate request using PKCS #10 [6] and sends it to the CA enveloped using CMS [3] (Section 3).

If the CA supports certificate renewal or update then 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. To renew or update an existing certificate, the client uses the RenewalReq or UpdateReq message (see Section 3.2) and signs it with the existing client certificate. The client SHOULD use a new keypair when requesting a new certificate, but MAY request a new certicate using the old keypair.

If the CA returns a CertRep (Section 3.2.2) message with status set to PENDING, the requester enters into polling mode by periodically sending a CertPoll (Section 3.2.3) PKI message to the CA, until the CA operator completes the manual authentication (approving or denying the request).

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

In general, the CA 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.

2.4.1. Client State Transitions

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

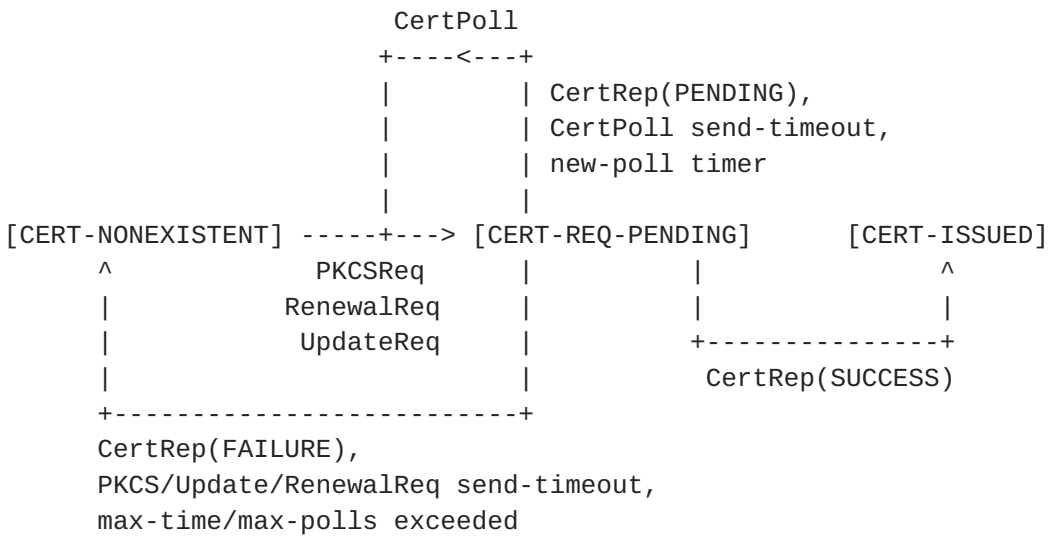


Figure 1: State Transition Diagram

The certificate issue process starts at the state CERT-NONEXISTENT.

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

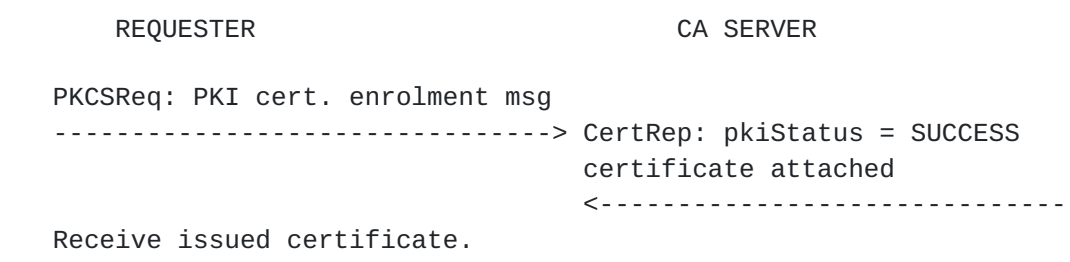
Receiving a CertRep message with pkiStatus set to SUCCESS changes the state to CERT-ISSUED.

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

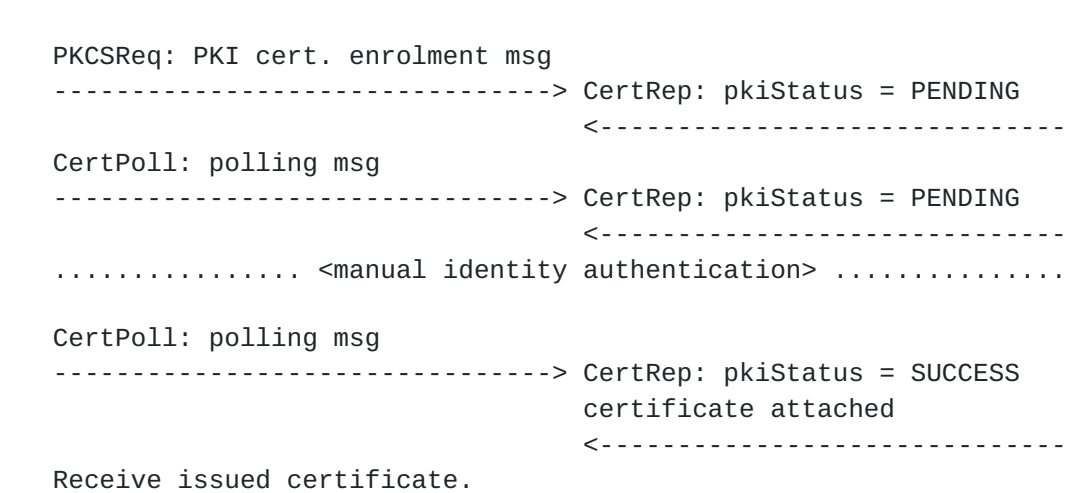
If the server sends back a CertRep message with pkiStatus set to PENDING, the requester will keep polling by sending a CertPoll message to the server, until either a CertRep message with status set to SUCCESS or FAILURE is received, or the maximum number of polls has been exceeded.

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-NONEXISTENT state, and the SCEP client can eventually initiate another enrolment request. It is important to note that, as long as the requester does not change its subject name or keys, the same transactionID may be used in the "new" transaction. This is important because based on this transactionID, the certification authority can recognize this as an existing transaction instead of a new one.

A successful transaction in automatic mode:



A successful transaction in manual mode:

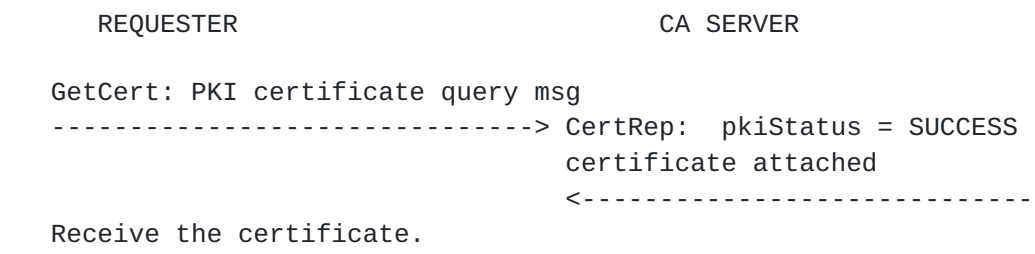


2.5. Certificate Access

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

certificate store access service, which may be achieved via HTTP [11] or LDAP.

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 enrolment 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 below.



2.6. CRL Access

SCEP clients MAY request a CRL via one of three methods:

1. If the CA supports CRL Distribution Points (CRLDPs) [7], then the CRL MAY be retrieved via the mechanism specified in the CRDLP.
2. If the CA supports HTTP [11], then the CRL MAY be retrieved via the AuthorityInfoAcces [7] location specified in the certificate.
3. Only if the CA does not support CRDLPs or HTTP access should a CRL query be composed by creating a GetCRL message consisting of the issuer name and serial number from the certificate whose revocation status is being queried.

The server SHOULD NOT support the GetCRL method because:

- It does not scale well due to the unnecessary cryptography (see Section 8).
- It requires the CA to be a high-availability service.
- Only limited information to determine the CRL scope is provided (see [7]).

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) in a degenerate certificates-only CMS [3] Signed-Data message (Section 3.3), as shown below.

2.7. Certificate Revocation

SCEP does not specify a method to request certificate revocation. In order to revoke a certificate, the requester must contact the CA using a non-SCEP defined mechanism.

2.8. Mandatory-to-Implement Functionality

At a minimum, all SCEP implementations compliant with this specification MUST support GetCACert (Section 4.1), PKCSReq (Section 3.2.1) (and its associated response messages), communication of binary data via HTTP POST (Section 5.1), and the AES and SHA-256 algorithms to secure pkiMessages (Section 3.1).

For historical reasons implementations MAY support communications of binary data via HTTP GET (Section 5.1), and the triple DES and SHA-1 algorithms to secure pkiMessages (Section 3.1).

3. SCEP Secure Message Objects

CMS [3] is a general enveloping mechanism that enables both signed and encrypted transmission of arbitrary data. SCEP messages that require confidentiality use two layers of CMS [3], as shown in Figure 2. 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 Enveloped-Data in Figure 2 is omitted.

```
pkiMessage {
  contentType = signedData
  content {
    pkcsPKIEnvelope {   -- Optional
      contentType = envelopedData
      content {
        recipientInfo
        contentType = data
        content {
          messageData   -- Typically PKCS #10 request
          }
        }
      }
    signerInfo {
      signedAttrs {
        transactionID
        messageType
        pkiStatus
        failInfo
        senderNonce
        recipientNonce
        }
      signature
      }
    }
  }
```

Figure 2: CMS Layering

When a particular SCEP message carries data, this data is carried in the messageData. CertRep messages will lack any signed content and consist only of a pkcsPKIEnvelope (Section 3.1.2).

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) of the Signed-Data. 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 a CMS [3] Signed-Data content type. The following restrictions apply:

- The contentType in contentInfo MUST be data ({pkcs-7 1}) as defined in CMS [3].
- 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.
- The SignerInfo MUST contain a set of authenticatedAttributes (see CMS [3] as well as Section 3.1.1 in this document).

At a minimum, all messages MUST contain the following authenticatedAttributes:

- A transactionID attribute (see Section 3.1.1.1).
- A messageType attribute (see Section 3.1.1.2).
- A senderNonce attribute (see Section 3.1.1.5).
- Any attributes required by CMS [3].

If the message is a response, it MUST also include the following authenticatedAttributes:

- A pkiStatus attribute (see Section 3.1.1.3).
- A recipientNonce attribute (see Section 3.1.1.5).

3.1.1. Signed Transaction Attributes

The following transaction attributes are encoded as authenticated attributes, and are carried, as specified in CMS [3], in the SignerInfo for this Signed-Data.

Attribute	Encoding	Comment
transactionID	PrintableString	Unique ID for this transaction as a text string
messageType	PrintableString	Decimal value as a numeric text string
pkiStatus	PrintableString	Decimal value as a numeric text string
failInfo	PrintableString	Decimal value as a numeric text string
senderNonce	OCTET STRING	Random nonce as a 16-byte binary data string
recipientNonce	OCTET STRING	Random nonce as a 16-byte binary data string

The OIDs used for these attributes are as follows:

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-transactionID	OBJECT_IDENTIFIER ::= {id-attributes transactionID(7)}
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)}

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 transactionID is a text 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 enrolment, encoded as a PrintableString.

One means of generating the transactionID is as a SHA-1, SHA-256, or SHA-512 hash of the public key value in the enrolment request when encoded as an X.509 SubjectPublicKeyInfo [7] (in other words the exact binary form in which it appears in both the request and the resulting certificate) and then coverting it into a text string using base64 encoding or ASCII hex digits. This allows the SCEP client to

automatically generate the same transactionID for any given public key. The SCEP protocol requires that transactionIDs be unique, so that subsequent polling queries can be matched with previous transactions.

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. For a non-enrolment message (for example GetCert and GetCRL), the transactionID SHOULD be some value 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. The following message types are defined:

- CertRep ("3") -- Response to certificate or CRL request.
- RenewalReq ("17") -- PKCS #10 [6] certificate request for renewal of an existing certificate.
- UpdateReq ("18") -- PKCS #10 [6] certificate request for update of a certificate issued by a different CA.
- PKCSReq ("19") -- PKCS #10 [6] certificate request.
- CertPoll ("20") -- Certificate polling in manual enrolment.
- GetCert ("21") -- Retrieve a certificate.
- GetCRL ("22") -- Retrieve a CRL.

Undefined message types are treated as an error.

3.1.1.3. pkiStatus

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

- SUCCESS ("0") -- request granted.
- FAILURE ("2") -- request rejected. When pkiStatus is FAILURE, the failInfo attribute, as defined in Section 3.1.1.4, MUST also be present.
- PENDING ("3") -- request pending for manual approval.

Undefined pkiStatus attributes are treated as an error.

3.1.1.4. failInfo

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

- badAlg ("0") -- Unrecognized or unsupported algorithm identifier.

- badMessageCheck ("1") -- integrity check failed.
- badRequest ("2") -- transaction not permitted or supported.
- badTime ("3") -- The signingTime attribute from the CMS [3] authenticatedAttributes was not sufficiently close to the system time (see Section 3.1.1.6).
- badCertId ("4") -- No certificate could be identified matching the provided criteria.

  [Question: Is there any demand for a free-form UTF8String attribute to explain what really went wrong? Trying to sort out an error when all you ever get back is the near-universal badRequest is almost impossible, adding a failInfoText attribute to address this could be quite useful since it would allow expressing information such as a failure to meet CA policy, or indeed anything more complex than "no go away"].

Undefined failInfo attributes are treated as an error.

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 sender sends a PKI message to a recipient, a senderNonce MUST be included in the message. The recipient MUST copy the senderNonce into the recipientNonce of the reply as a proof of liveliness. The original sender MUST verify that the recipientNonce of the reply matches the senderNonce it sent in the request. If the nonce does not match, the message MUST be rejected.

[Question: What does this do for polling? Polling messages can get lost so nonces will go out of sync, is there a need to chain XXXReqs to polls via nonces? If not, why do we have two nonces?].

3.1.2. SCEP pkcsPKIEnvelope

The information portion of a SCEP message is carried inside an Enveloped-Data content type, as defined in CMS [3], with the following restrictions:

- contentType in encryptedContentInfo MUST be data ({pkcs-7 1}) as defined in CMS [3].
- encryptedContent MUST be the SCEP message being transported (see Section 4), and must match the messageType authenticated Attribute in the pkiMessage.

The CMS [3] content-encryption key is encrypted using the public key of the recipient of the message, i.e. 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/RenewalReq/UpdateReq

The messageData for this type consists of a PKCS #10 [6] Certification Request. The certification request MUST contain at least the following items:

- The subject Distinguished Name.
- The subject public key.
- For a PKCSReq and if authorisation based on a password is being used, a challengePassword attribute.

In addition to the authenticatedAttributes required for a valid CMS [3] message, the 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, RenewalReq, or UpdateReq as appropriate.
- 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 public key.

3.2.2. CertRep

The messageData for this type consists of a degenerate certificates-only CMS [3] Signed-Data message (Section 3.3). The exact content required for the reply depends on the type of request this message is a reply to. They are detailed in Section 3.2.2.1 and in Section 4.

In addition to the authenticatedAttributes required for a valid CMS [3], 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 that this is a response 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.

Note that although it may appear that the senderNonce serves no purpose in this message, it is required if the CertRep contains a PENDING status since the nonce will be used in subsequent polling operations.

3.2.2.1. CertRep SUCCESS

When the pkiStatus attribute is set to SUCCESS, the messageData for this message consists of a degenerate certificates-only CMS [3] Signed-Data message (Section 3.3). The content of this degenerate certificates-only Signed-Data depends on what the original request was, as outlined below.

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 leaf certificate. The reply MUST NOT contain a CRL.
RenewalReq	Same as PKCSReq
UpdateReq	Same as PKCSReq
CertPoll	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 leaf certificate. The reply MUST NOT contain a CRL.
GetCRL	The reply MUST contain the CRL in the crls field of the Signed-Data. The reply MUST NOT contain a certificate.

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. CertPoll (GetCertInitial)

This message is used for certificate polling. For unknown reasons it was referred to as "GetCertInitial" in earlier drafts. The messageData for this type consists of an IssuerAndSubject:

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

The issuer is set to the subjectName of the CA (in other words the intended issuerName of the certificate that's being requested). The Subject is set to the subjectName used when requesting the certificate.

In addition to the authenticatedAttributes required for a valid CMS [3], this pkiMessage MUST include the following attributes:

- The same transactionID (Section 3.1.1.1) attribute from the original PKCSReq/RenewalReq/UpdateReq message.
- A messageType (Section 3.1.1.2) attribute set to CertPoll.
- A senderNonce (Section 3.1.1.5) attribute.
- A recipientNonce attribute (Section 3.1.1.5) copied from the senderNonce from the request that this is a response to.

3.2.4. GetCert

The messageData for this type consists of an IssuerAndSerialNumber as defined in CMS [3] which uniquely identifies the certificate being requested.

In addition to the authenticatedAttributes required for a valid CMS [3], 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 IssuerAndSerialNumber as defined in CMS [3] containing the issuer name and serial number of the certificate whose revocation status is being checked.

In addition to the authenticatedAttributes required for a valid CMS [3], 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 GetCRL.
- A senderNonce (Section 3.1.1.5) attribute.

3.3. Degenerate certificates-only CMS Signed-Data

CMS [3] includes a degenerate case of the CMS [3] Signed-Data content type, in which there are no signers. The use of such a degenerate case is to disseminate certificates and CRLs. 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.

3.4. CA Capabilities

In order to provide support for future enhancements to the protocol, CAs SHOULD implement the GetCACaps message to allow clients to query which functionality is available from the CA.

3.4.1. GetCACaps HTTP Message Format

This message requests capabilities from a CA, with the format:

"GET" CGI-PATH CGI-PROG "?operation=GetCACaps"

with the message components as described in Section 5. The response is a list of text capabilities, as defined in Section 3.4.2. CA servers SHOULD support the GetCACaps message and MUST support it when they implement any extended functonality beyond the mandatory-to-implement basics Section 2.8.

3.4.2. 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
"AES"	CA Supports the AES encryption algorithm.
"DES3"	CA Supports the triple DES encryption algorithm.
"GetNextCACert"	CA Supports the GetNextCACert message.
"POSTPKIOperation"	PKIOPeration messages may be sent via HTTP POST.
"Renewal"	CA Supports the Renewal CA operation.
"SHA-1"	CA Supports the SHA-1 hashing algorithm.
"SHA-256"	CA Supports the SHA-256 hashing algorithm.
"SHA-512"	CA Supports the SHA-512 hashing algorithm.
"Update"	CA Supports the Update CA operation.

The client SHOULD use SHA-256 or SHA-512 in preference to SHA-1 hashing, and AES in preference to triple DES if they are supported by the CA.

Announcing some of these capabilities is redundant since they're required as mandatory-to-implement functionality (see Section 2.8), but it may be useful to announce them in order to deal with old implementations that would otherwise default to obsolete, insecure algorithms and mechanisms.

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.

(Note that at least one widely-deployed server implementation supports several of the above operations but doesn't support the GetCACaps message to indicate that it supports them. This means that the equivalent of GetCACaps must be performed through server fingerprinting, which can be done using the ID string "Microsoft-IIS").

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
```

might return:

```
AES
SHA-256
GetNextCACert
POSTPKIOperation
```

This means that the CA supports modern crypto algorithms, the GetNextCACert message, and allows PKIOperation messages (PKCSReq/RenewalReq/UpdateReq, GetCert, CertPoll, ...) to be sent using HTTP POST.

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).

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

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 server MUST indicate which response it is sending via the transport protocol used (see Section 5.2.1). If the requester does not have a certificate path to a trust anchor certificate, the SHA-1, SHA-256, or SHA-512 fingerprint of the returned CA certificate (communicated via out- of- band means) may be used to verify it.

All returned certificates MUST conform to PKIX [7].

4.1.1.1. CA Certificate Response Message Format

If the server does not have any intermediate CA certificates, the response consists of a single X.509 CA certificate.

4.1.1.2. CA Certificate Chain Response Message Format

If the server has intermediate CA certificates, the response consists of a degenerate certificates-only CMS [3] Signed-Data (Section 3.3) containing the certificates, with the intermediate CA certificate(s) as the leaf certificate(s).

4.2. Certificate Enrolment/Renewal/Update

A PKCSReq/RenewalReq/UpdateReq (Section 3.2.1) message is used to perform a certificate enrolment, renewal, or update 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 certificate.

Postcondition: The requester receives 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 Enrolment/Renewal/Update 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 CertPoll messages (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 some preconfigured time limit for polling or maximum number of polls is exceeded.

CertPoll messages exchanged during the polling period MUST carry the same transactionID attribute as the previous PKCSReq/RenewalReq/ UpdateReq. A server receiving a CertPoll for which it does not have a matching PKCSReq/RenewalReq/UpdateReq 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/RenewalReq/UpdateReq) to identify the polled certificate request. In theory there can be multiple outstanding requests from one requester (for example, if different keys and different key-usages were used to request multiple certificates), so the transactionID must also be included to disambiguate between multiple requests. In practice however it's safer for the requester to not have multiple requests outstanding at any one time, since this tends to confuse some servers.

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 CertPoll 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 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 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 can request a CRL from the SCEP server as described in Section 2.6.

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.

4.6. Get Next Certification Authority Certificate

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. 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).

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

4.6.1. Get Next CA Response Message Format

The response consists of a Signed-Data CMS [3], signed by the current CA signing key. Clients MUST validate the signature on the the Signed-Data CMS [3] before accepting any of its contents.

The content of the Signed-Data CMS [3] message is a degenerate certificates-only Signed-Data (Section 3.3) message containing the new CA certificate(s) as defined in Section 5.2.1.1.2, to be used when the current CA certificate expires.

If the CA 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 intermediate CA certificates in this response, clients MUST check that these certificates are signed by the CA, and MUST check authorization of these intermediate CA certificates (see Section 2.1.2).

5. SCEP Transport

HTTP [4] is used as the transport protocol for SCEP Message Objects.

5.1. HTTP GET and POST Message Formats

SCEP uses the HTTP "GET" and "POST" messages to exchange information with the CA. The following defines the syntax of a HTTP GET and POST messages sent from a requester to a certification authority server:

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

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, or ignore both if it's not issuing certificates via a web server. Typically, setting CGI-PATH/CGI-PROG to "/cgi-bin/pkiclient.exe" will satisfy most servers.
- OPERATION depends on the SCEP transaction and is defined in the following sections.

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

Early SCEP drafts performed all communications via "GET" messages, including non-idempotent ones that should have been sent via "POST" messages. This has caused problems because of the way that the (supposedly) idempotent GET interacts with caches and proxies, and because the extremely large GET requests created by encoding CMS messages may be truncated in transit. These issues are typically not visible when testing on a LAN, but crop up during deployment over WANs. If the remote CA supports it, any of the CMS [3]-encoded SCEP messages SHOULD be sent via HTTP POST instead of HTTP GET. This is allowed for any SCEP message except GetCACert, GetNextCACert, or GetCACaps, and avoids the need for base64- and URL-encoding that's required for GET messaging. The client can verify that the CA supports SCEP messages via POST by looking for the "POSTPKIOperation" capability (See Section 3.4.2).

If your client or server uses HTTP GET and encounters HTTP-related problems such as messages being truncated, seeing errors such as HTTP 414 ("Request URI too long"), or simply having the message not sent/ received at all, when standard requests to the server (for example via a web browser) work, then this is a symptom of the problematic use of HTTP GET. The solution to this problem is typically to move to HTTP POST instead. In addition when using GET it's recommended to test your implementation over the public internet from as many locations as possible to determine whether the use of GET will cause problems with communications.

When using GET messages to communicate binary data, base64 encoding as specified in [2] MUST be used. The base64 encoded data is distinct from "base64url" and may contain URI reserved characters, thus it MUST be escaped as specified in [8] in addition to being bas64 encoded.

5.1.1. Response Message Format

For each GET or POST operation, the CA 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".

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 enrolment, and the case where an intermediate CA exists and the requester communicates with this CA during the enrolment.

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 an X.509 CA certificate, as defined in Section 4.1.1.1:

"Content-Type:application/x-x509-ca-cert"

<binary X.509>

5.2.1.1.2. CA Certificate Chain Response

The response will have a Content-Type of "application/x-x509-ca-ra-cert". Note that this designation is used for historical reasons due to its use in older SCEP drafts, no special meaning should be attached to the label itself.

The body of this response consists of a degenerate certificates-only CMS [3] Signed-Data (Section 3.3) message containing both CA and intermediate CA certificates, as defined in Section 4.1.1.2:

"Content-Type:application/x-x509-ca-ra-cert"

<binary CMS>

5.2.2. PKCSReq/RenewalReq/UpdateReq

The OPERATION MAY be set to "PKIOperation". When used with HTTP POST, the only OPERATION possible is "PKIOperation", so many servers don't check this values or even notice its absence.

The MESSAGE consists of a PKCSReq, RenewalReq, or UpdateReq SCEP message. When implemented using HTTP POST this might look as follows:

```
POST /cgi-bin/pkiclient.exe?operation=PKIOperation HTTP/1.1
Content-Length: <length of data>

<binary CMS data>
```

When implemented using HTTP GET this might look as follows:

```
GET /cgi-bin/pkiclient.exe?operation=PKIOperation& \
message=MIAGCSqGSIb3DQEHA6CAMIACAQAxgDCBzAIBADB2MG \
IxETAPBgNVBAcTCE......AAAAAA== HTTP/1.1
```

5.2.2.1. PKCSReq/RenewalReq/UpdateReq Response

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

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

```
"Content-Type:application/x-pki-message"

<binary CertRep msg>
```

5.2.3. CertPoll

The OPERATION MUST be set to "PKIOperation". The MESSAGE consists of a CertPoll SCEP message.

5.2.3.1. CertPoll Response

The body of this response consists of a 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 GetCert SCEP message.

5.2.4.1. GetCert Response

The body of this response consists of a 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 GetCRL SCEP message.

5.2.5.1. GetCRL Response

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

5.2.6. GetNextCACert

The OPERATION MUST be set to "GetNextCACert".

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 Signed-Data CMS [3], 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"

<binary CMS>

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, Max Pritikin, Jan Vilhuber, 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 earlier 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:

- Subvert the public key/identity binding from that intended.
- Discover the identity information in the enrolment requests and issued certificates.

Here an adversary is any entity other than the requester and the CA participating in the protocol. The adversary is computationally limited, but that can manipulate data during transmission (that is, can act as a MITM). The precise meaning of 'computationally limited' depends on the implementer's choice of one-way hash functions and cryptographic algorithms.

The first and second goals are met through the use of CMS [3] and PKCS #10 [6] encryption and digital signatures using authenticated public keys. The CA's public key is authenticated via out-of-band means such as the checking of the CA fingerprint, as specified in Section 2.1.2, and the SCEP client's public key is authenticated through manual or pre-shared secret authentication, as specified in Section 2.2.

The motivation of the first security goal is straightforward. The motivation for the second security goal is to protect the identity information in the enrolment 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 (and CRL) signing exclusively, rather than data signing or encryption. The SCEP protocol, however, uses the CA private key to both encrypt and sign CMS [3] transport messages. This is generally considered undesirable, as it widens the possibility of an implementation weakness, and provides:

- Another place that the private key must be used (and hence is slightly more vulnerable to exposure).
- Another place where a side channel attack (say, timing or power analysis) might be used.
- Another place that the attacker might somehow insert their own data and get it signed by the CA's private key (note that this issue is purely theoretical, since the CMS data signed by the CA is nothing remotely like a certificate and couldn't be passed off as such).

8.3. Challenge Password

The challengePassword sent in the PKCS #10 enrolment 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 enrolment 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. Transaction ID

CAs 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. A well-written CA 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.

8.6. GetCACaps Issues

The GetCACaps response is not signed. This allows an attacker to perform downgrade attacks on the cryptographic capabilities of the client/CA exchange.

8.7. 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 CRLDPs as well as other ways of retrieving certificates such as HTTP access and LDAP are recommended for CRL access.

8.8. GetNextCACert

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. References

9.1. Normative References

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

[2] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[3] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[4] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

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

[7] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 5280, May 2008.

[8] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

9.2. Informative References

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

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

[11] Gutmann, P., "Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP", RFC 4387, February 2006.

[12] Alighieri, D., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, March 1300.

[13] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[14] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Appendix A. 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 enrolment transaction is uniquely associated with a transactionID (carried in the transactionID signed attribute (see Section 3.1.1.1). Because the enrolment transaction could be interrupted by various errors, including network connection errors or client reboot, the SCEP client generates a fixed transaction identifier as specified in Section 3.1.1.1 which is included in the PKCSReq/RenewalReq/UpdateReq. If the CA returns a response of

PENDING, the requester will poll by periodically sending a CertPoll 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 enrolment transaction.

If the client times out or reboots, the client administrator will start another transaction with the same key pair. The second enrolment will have the same transactionID. At the server side, instead of accepting the PKCSReq/RenewalReq/UpdateReq as a new request, it can respond as if another CertPoll message had been sent with that transaction ID. The second PKCSReq/RenewalReq/UpdateReq should be taken as a resynchronization message to allow the process to resume as the same transaction.

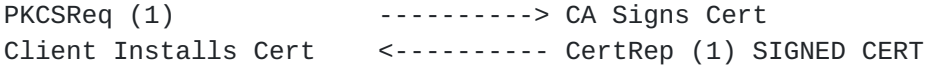
The following gives several examples of client to CA transactions.

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 transactionID 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 Enrolment Case: no manual authentication



Successful Enrolment Case: manual authentication required

```
PKCSReq (10)            ----------> Cert Request goes into Queue
Client Polls            <---------- CertRep (10) PENDING
CertPoll (10)           ----------> Still pending
Client Polls            <---------- CertRep (10) PENDING
CertPoll (10)           ----------> Still pending
Client Polls            <---------- CertRep (10) PENDING
CertPoll (10)           ----------> Still pending
Client Polls            <---------- CertRep (10) PENDING
CertPoll (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
CertPoll (3)          ----------> Still pending
                      <---------- CertRep (3) PENDING
CertPoll (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
```

CA certificate rollover case:

```
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
existing cert expires.            pair
```

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

Appendix B. Background Notes

This specification has spent more than fifteen years in the draft stage. Its original goal, provisioning IPsec routers with RSA certificates, has long since changed to general device/embedded system/IoT use. To fit this role, extra features were bolted on in a haphazard manner through the addition of a growing list of appendices and by inserting additional, often conflicting, paragraphs in various locations in the body text. Since existing features were never updated as newer ones were added, the specification accumulated large amounts of historical baggage over time. If OpenPGP was described as "a museum of 1990s crypto" then the SCEP draft was its graveyard.

About five years ago the specification, which even at that point had seen only sporadic re-posts of the existing document, was more or less abandoned by its original sponsors. Due to its widespread use in large segments of the industry, the specification was rebooted in 2015, cleaning up fifteen years of accumulated cruft, fixing errors, clarifying ambiguities, and bringing the algorithms and standards used into the current century (prior to the update, the de-facto lowest-common denominator algorithms used for interoperability were the forty-year-old single DES and broken MD5 hash algorithms).

Other changes include:

- Resolved contradictions in the text, for example a requirement given as a MUST in one paragraph and a SHOULD in the next, a MUST NOT in one paragraph and a MAY a few paragraphs later, a SHOULD NOT contradicted later by a MAY, and so on.
- Merged several later fragmentary addenda placed in appendices (for example the handling of certificate renewal and update) with the body of the text.
- Updated the algorithms to ones dating from at least this century.
- Did the same for normative references to other standards.
- Corrected incorrect references to other standards, e.g. IssuerAndSerial -> IssuerAndSerialNumber.
- Corrected errors such as a statement that when both signature and encryption certificates existed, the signature certificate was used for encryption.
- Condensed redundant discussions of the same topic spread across multiple sections into a single location. For example the description of intermediate CA handling previously existed in three different locations, with slightly different reqirements in each one.
- Relaxed some requirements that didn't serve any obvious purpose and that major implementations didn't seem to be enforcing. For example the requirement that the self-signed certificate used with a request MUST contain a subject name that matched the one in the PKCS #10 request was relaxed to a SHOULD because a number of implementations either ignored the issue entirely or at worst performed some minor action like creating a log entry after which they continued anyway.
- Clarified sections that were unclear or even made no sense, for example the requirement for a "hash on the public key [sic]" encoded as a PrintableString.
- Renamed "RA certificates" to "intermediate CA certificates". The original document at some point added mention of RA certificates without specifying how the client was determine that an RA was in use, how the RA operations were identified in the protocol, or how it was used. It's unclear whether what was meant was a true RA or merely an intermediate CA, as opposed to the default practice of

having certificates issued directly from a single root CA certificate. This update uses the term "intermediate CA certificates", since this seems to have been the original intent of the text.

- Clarified certificate renewal and update. These represent a capability that was bolted onto the original protocol with (at best) vaguely-defined semantics, including a requirement by the server to guess whether a particular request was a renewal or not (updates were even more vaguely defined). In response to developer feedback that they either avoided renewal/update entirely because of this uncertainty or hardcoded in particular behaviour on a per-server basis, this specification explicitly identifies renewal and update requests as such, and provides proper semantics for both. Note that this is still a work in progress due to the lack of clarity of the original spec in this area, see some of the questions inline with the text.
- Removed the discussion in the security considerations of revocation issues, since SCEP doesn't support revocation.

Authors' Addresses

Peter Gutmann
University of Auckland
Department of Computer Science
Auckland
New Zealand

Email: pgut001@cs.auckland.ac.nz

Max Pritikin
Cisco Systems, Inc