PKIX Working Group
Internet Draft

Michael Myers (VeriSign)
Xiaoyi Liu (Cisco)
Barbara Fox (Microsoft)
Jeff Weinstein (Netscape)

expires in six months

August 1998

Internet X.509 Certificate Management Messages over CMS
<draft-ietf-pkix-cmc-01.txt>

1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Not that other groups MAY also distribut working documents as Internet-Drafts.

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."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munari.oz.au Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

2. Abstract

This document defines the means by which PKI clients and servers may exchange PKI messages when using the Cryptographic Message Syntax [CMS] as a transaction envelope. It extends concepts established in the draft [CRS] version of this material by accommodating external specification of message bodies in the Certificate Management Message Formats [CMMF] and Certificate Request Message Format [CRMF] documents. Mandatory requirements are established which facilitate automated interoperability between a wide variety of PKI clients and servers or services. Optional features are defined to address more localized needs.

This draft is being discussed on the ''ietf-pkix'' mailing list. To subscribe, send a message to ietf-pkix-request@tandem.com with the single word "subscribe" in the body of the message.

3. Protocol Overview

This document defines a protocol by which PKI-specific messages can be encapulsated within the CMS security framework. CMMF message bodies secured by CMS are encapsulated either as PKIData content type or Data content type. This encapsulation is in turn ultimately encapsulated by a SignedData envelope. The authenticatedAttributes element of SignedData is used to carry information useful to recipient processing of CMS-encapsulated CMMF message bodies. These are collectively referred to as "Service Indicators".

Two such indicators are MessageType (an attribute unique to this specification) and ContentType (defined by [CMS]).

Processing systems, whether client or server:

1) Detects the presence of a PKI message.
2) Examines the value of MessageType;
3) Locates the PKI message within the CMS envelope;
4) Parses the content of PKI message in accordance with MessageType; and
5) Take action as appropriate to the message body content.

Successful processing of the message may depend in part upon other authenticated attributes within the SignedData envelope.

3.1 Transactions

A transaction is composed of the exchange of request-response message pairs between a server and a client. This document establishes requirements on four such transactions. These are:

1. Certification of a public key;
2. Query on status of certificate request;
3. Certificate and CRL retrieval;
4. Certificate revocation.

A public key certification request is formed either as a PKCSReq object or CRMF; the [CMMF] document identifies both. PKCSReq enables use of PKCS10-based certificate requests while CRMF anticipates a richer set of requirements than can be met by PKCS10. The corresponding response is either by CertResponse or CertRepContent as defined in CMMF. The CertResponse message body incorporates the familiar PKCS7 degenerate signedData mode of certificate delivery while CertRepContent provides equivalent capability and enables CA generation of subscriber key pairs.

The certificate and CRL retrieval request mechanism is used to assist state recovery within resource-constrained PKI clients. This may be the case, for example, within low-end IP routers implementing IPSEC which by reasons of design do not retain such data in non-volatile memory.

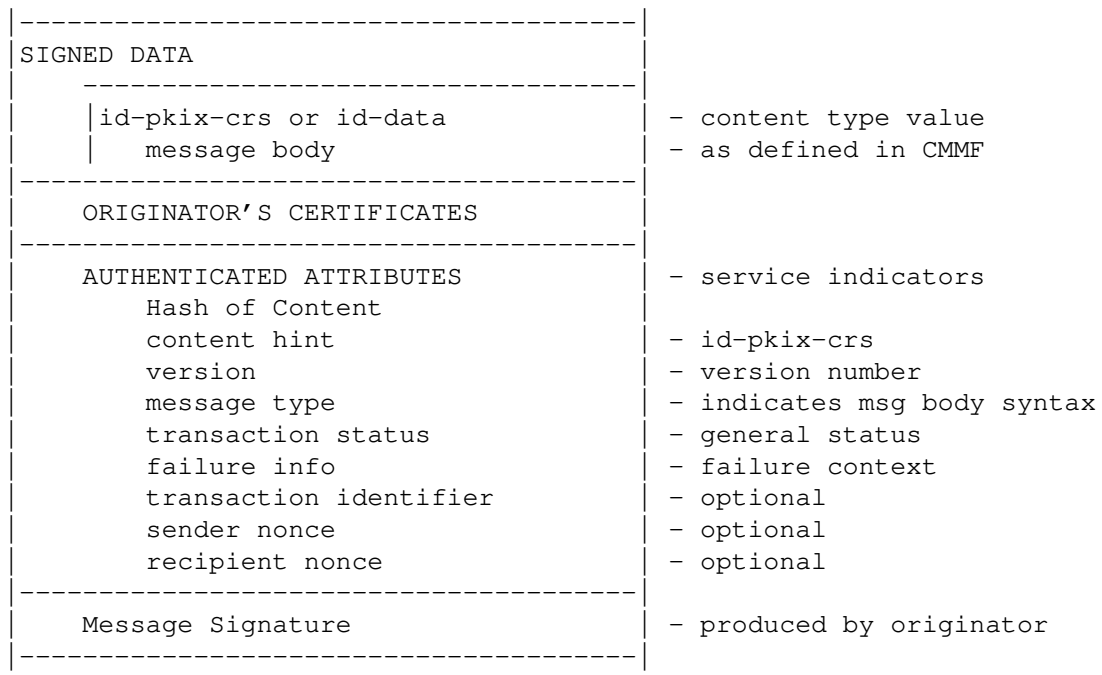
The certificate request status query can be used as a simple means to maintain synchronicity with a certificate server during the certificate production process. This message would typically be used in a situation where the certificate requesting system holds itself in a wait state during the processing of certificate request.

The revocation request transaction enables programmatic processing of revocation requests. It further provides for non-cryptographic authentication of the requester in the event that the key being revoked would otherwise be relied upon to authenticate the originator of a revocation message. This capability is useful in the event of a key compromise.

3.2 Interior Encapsulation of PKI Messages

The inner-most content type used encapsulate PKIData (i.e. CRMF or CMMF

message bodies) is either id-pkix-crs or id-data. PKIData is used with SignedData to define a transaction-based protocol. EnvelopedData may be used in conjunction with SignedData to provide privacy. The following figure illustrates the relationship between PKIData and SignedData in the case of an unencrypted message (section 4.3 discusses encryption options):



Section 3 of [CMS] establishes the following general syntax for CMS content types:

```
ContentInfo ::= SEQUENCE {
        contentType ContentType,
        content [0] EXPLICIT ANY DEFINED BY contentType }

ContentType ::= OBJECT IDENTIFIER
```

For the purposes of the CMC protocol, the content field of ContentInfo will contain PKIData. The value of contentType of a ContentInfo containing PKIData SHALL be constrained as follows. Applications that use CMS content types to detect and route content to appropriate handling logic SHOULD use the value id-pkix-crs to identify the presence of PKIData. If id-pkix-crs is not used, id-data SHALL be used. In either case, the specific syntax contained within PKIData is indicated by the MessageType attribute in an encapsulated SignedData content type.

Transactions can be identified and tracked using a transaction identifier. If used, clients generate transaction identifiers and retain their value until the server responds with a message that completes the transaction. Servers correspondingly include received

transaction identifiers in the response. Section 5.5 establishes requirements on the use of TransactionID.

Replay protection is supported through the use of sender and recipient nonces. If used, clients generate a nonce value and include it in the

request as a sender nonce. Servers return this value as recipient nonce along their own value for sender nonce.

This specification makes no assumptions about the underlying transport mechanism. The use of CMS is not meant to imply an email-based transport.

Requests and responses are composed of a message body and one or more Service Indicators. Service Indicators are encoded as a set of authenticated attributes of a CMS SignedData construction. The message digest of message body is then signed together with the Service Indicators using the message originator's private signing key, producing the message signature.

4. Protocol Requirements

4.1 PKCS7 Interoperability

CMS differs from PKCS7 in that it:

- adds to EnvelopedData an OPTIONAL originatorInfo field preceding recipientInfo;
- replaces the issuerAndSerial field of recipientInfos with a CHOICE of alternative recipient key identification mechanisms.

Clients MAY include the optional OriginatorInfo field of the CMS EnvelopedData syntax when submitting PKI transaction requests. If the intended recipient is unable to receive this optional syntax, an error response message SHALL be generated per Section 4.3.1.

Clients SHALL be capable of receiving and servers SHALL be capable of processing the issuerAndSerialNumber CHOICE of the rid (recipient identifer) syntax for RecipientInfos. The corresponding RecipientKeyIdentifier is optional within this specification.

As noted in [CMS], if RecipientKeyIdentifier is used, the value for version in EnvelopedData SHALL be 2; if issuerAndSerialNumber CHOICE is used, the value for version SHALL be 0.

The specification of CMS ensures that EnvelopedData productions with a version of 0 will successfully interoperate with systems implemented in accordance with PKCS7.

4.2 Mandatory and Optional Algorithms

Clients and servers SHALL be capable of producing and processing message signatures using the Digital Signature Algorithm [DSA]. DSA signatures SHALL be indicated by the DSA AlgorithmIdentifier value specified in section 7.2.2 of PKIXCERT. Clients and servers SHOULD

also be capable of producing and processing RSA signatures as specified in section 7.2.1 of PKIXCERT.

Clients and servers SHALL be capable of protecting and accessing message encryption keys using the Diffie-Hellman (D-H) key exchange algorithm. D-H protection SHALL be indicated by the D-H AlgorithmIdentifier value specified in CMS. Clients and servers

SHOULD also be capable of producing and processing RSA key transport. When used for PKI messages, RSA key transport SHALL be indicated as specified in section 7.2.1 of PKIXCERT.

4.3 Use of EnvelopedData

Two options exist with respect to the use of EnvelopedData. One usage produces an encrypted message body encapsulated by SignedData. The other option is the inverse of this relationship, as the following figure illustrates.

```
Option 1                          Option 2
--------                          --------
SignedData                        EnvelopedData
   EnvelopedData                     SignedData
      PKIData                           PKIData
         <PKI message body>                <PKI message body>
```

The second option benefits organizations that wish to protect against the leakage of sensitive data via the cleartext SignedData envelope.

Message bodies MAY be encrypted or transmitted in the clear. Support SHALL be provided for encryption option 1 and SHOULD be provided for both.

4.4 Requirements on Message Construction

The exact processing sequence used to construct a message could vary by application. The results SHALL however be structurally equivalent to the following procedure:

A [CMMF] message body is constructed in accordance with this specification and any additional requirements asserted in [CMMF] regarding a given message body. This message body is then identified as a contentType as defined in section 4.3, yielding PKIData.

If PKIData is to be encrypted, then either an EnvelopedData content type is constructed which contains the PKIData--which is in turn enveloped by SignedData as an outermost CMS envelope--or a SignedData content type is constructed which is subsequently encapsulated by EnvelopedData as the outermost CMS envelope. The decision on which to use is optional to the implementor, although section 5.3 establishes the requirement that all implementors shall at least enable the former (i.e. SignedData <- Enveloped Data <- PKIData).

If PKIData is to be transmitted in cleartext form, then a SignedData is contructed with PKIData as the content and id-signedData as the contentType of the outermost CMS envelope.

The SignerInfos portion of SignedData carries one or more Service Indicators as authenticatedAttributes. In all cases the presence of a particular message body type SHALL be indicated by value of the messageType Service Indicator. In addition, a value of id-pkix-crs as defined in [PKIXCERT] MAY be assigned to a contentHint authenticated

attribute as defined in [ESS].

The value of authenticatedAttributes is hashed using the algorithm specified by digestAlgorithm, signed using the message originator's private key corresponding to digestEncryptionAlgorithm, the result encoded as an OCTET STRING and assigned to the encryptedDigest field of SignedData.

The following illustrates the intended relationship between the relevant syntactic elements:

Encryption Option 1: (signedData <- envelopedData <- pkiData)

```
signedData.encapContentInfo.eContentType            <- id-envelopedData
signedData.authenticatedAttributes.contentHint      <- id-pkix-crs
signedData.authenticatedAttributes.messageType      <- CMMF message type
envelopedData.encryptedContentInfo.contentType      <- id-pkiData
envelopedData.encryptedContentInfo.encryptedContent <- encrypted CMMF
                                                       message body
```

Encryption option 2: (envelopedData <- signedData <- pkiData)

```
envelopedData.encryptedContentInfo.contentType      <- id-signedData
envelopedData.encryptedContentInfo.encryptedContent <- encr. signedData
signedData.authenticatedAttributes.contentHint      <- id-pkix-crs
signedData.authenticatedAttributes.messageType      <- CMMF message type
signedData.encapContentInfo.eContentType            <- id-pkiData
signedData.encapContentInfo.eContent                <- CMMF message body
```

The cleartext version is: (signedData <- pkiData)

```
signedData.authenticatedAttributes.contentHint      <- id-pkix-crs
signedData.authenticatedAttributes.messageType      <- CMMF message type
signedData.encapContentInfo.eContentType            <- id-pkiData
signedData.encapContentInfo.eContent                <- CMMF message body
```

4.5 Service Indicators

The following Service Indicators are defined by this specification.

- version
- messageType
- pkiStatus
- failinfo
- transactionId

- senderNonce
- recipientNonce

In addition to the above, [CMS] requires inclusion of the following:

- A content-type attribute having as its value the content type of the ContentInfo value being signed.

- A message-digest attribute, having as its value the message digest of the content.

In addition, clients MAY include provisions for SigningTime, counterSignature and contentHint attributes. Servers SHOULD be capable of accepting PKIData messages containing such attributes.

Each Service Indicator is uniquely identified by an Object Identifier. Processing systems would first detect the OID and process the corresponding service indicator value prior to processing the message body. PKIXCERT establishes a registration arc for objects associated with certificate management protocols. The value of id-it is imported from that reference. This specification extends id-it as follows:

```
id-it OBJECT IDENTIFIER  ::= { id-pkix 4 }   -- imported from PKIX
id-si OBJECT IDENTIFIER  ::= { id-it 1   }   -- cmc service indicators

-- service indicators

id-si-version          OBJECT IDENTIFIER ::= { id-si 1 }
id-si-transactionID    OBJECT IDENTIFIER ::= { id-si 2 }
id-si-messageType      OBJECT IDENTIFIER ::= { id-si 3 }
id-si-pkiStatus        OBJECT IDENTIFIER ::= { id-si 4 }
id-si-failInfo         OBJECT IDENTIFIER ::= { id-si 5 }
id-si-senderNonce      OBJECT IDENTIFIER ::= { id-si 6 }
id-si-recipientNonce   OBJECT IDENTIFIER ::= { id-si 7 }
```

The corresponding value syntax for each is:

Service Indicator	Syntax
version	INTEGER
messageType	INTEGER
pkiStatus	INTEGER
failInfo	INTEGER
TransactionId	INTEGER
senderNonce	OCTET STRING
recipientNonce	OCTET STRING

If version is absent, a value of 0 SHALL be assumed. This draft specifies version 1.

The messageType service indicator identifies the syntax carried in the message body. Every message SHALL include a value for messageType appropriate to the message.

The pkiStatus service indicator is used to convey information relevant to a requested operation. This service indicator SHALL be included in every message.

The failInfo service indicator conveys information relevant to the interpretation of a failure condition. This service indicator is mandatory in every message.

If the status in the response is FAILURE, then the failinfo service indicator SHALL contain one of the failure reasons defined in Section 5.4.1 "Specific Values". Additional failure reasons MAY be defined for environments with a need.

If additional transaction management or replay protection is desired, transactionID, senderNonce and recipientNonce MAY be implemented.

The transactionId service indicator identifies a given transaction. It is used between client and server to manage the state of an operation. It MAY be included in service request messages. If included, responses SHALL include the transmitted value. Correspondingly, clients SHOULD implement and recognize transactionID while servers or services SHALL implement and recognize transactionID.

The senderNonce and recipientNonce service indicator can be used to provide application-level replay prevention. They MAY be included in service request messages. Originating messages SHALL include only a value for senderNonce. If included in originating messages, responses SHALL include the transmitted value of the previously received senderNonce as recipientNonce and include a value for senderNonce.

If nonces are used, in the first message of a transaction, no recipient-Nonce is transmitted; a senderNonce is instantiated by the message originator and retained for later reference.

If a transaction originator includes a value for the senderNonce service indicator, responses SHALL include this value as a value for recipient-Nonce AND include a value for the SenderNonce service indicator.

Upon receipt by the transaction originator of response containing a value for recipientNonce, the originator compares the value of recipientNonce to its retained value. If the values match, the message can be accepted for further security processing. The received value for senderNonce is also retained for inclusion in the next message associated with the same transaction.

4.5.1 Specific Values

This specification establishes requirements regarding the implementation of message formats defined in [CMMF]. The following values for MessageType service indicator SHALL be used to identify the indicated message body type:

Message	MessageType Value	Legacy Value
PKCSReq	1	19
CertReqMessages	2	
CertRep	3	
CertRepContent	4	
GetCert	5	21
GetCertInitial	6	
GetCRL	7	22
RevRequest	8	
RevReqContent	9	
RevRepContent	10	
CAKeyUpdAnnContent	11	
CertAnnContent	12	
CRLAnnContent	13	

```
RevAnnContent          14
KeyRecRepContent       15
```

The pkiStatus field may take on any one of the following values:

Status	pkiStatus value
SUCCESS	0
PENDING	1
FAILURE	2

The following values for failInfo are defined for the identified context.

```
BADALG            0  -- Unrecognized or unsupported algorithm
BADMESSAGECHECK   1  -- integrity check failed
BADREQUEST        2  -- transaction not permitted or supported
BADTIME           3  -- Message time field was not sufficiently close
                        -- to the system time
BADCERTID         4  -- No certificate could be identified matching
                        -- the provided criteria
UNSUPPORTEDEXT    5  -- A requested X.509 extension is not supported
                        -- by the recipient CA.
BADTRANSID        6  -- Unrecognized transactionID
```

Additional contextual information regarding failure modes and reasons may be provided within specific message bodies.

4.6 Additional Requirements on Use of Transaction ID

Upon receipt at a server or service of an initiating message containing a TransactionID authenticated attribute, the value of TransactionID is retained during message processing and included in the response message. In general, the server may at that point delete retention of the TransactionID from local memory. However, when a server issues a PENDING status against a request, the server SHALL retain the value of TransactionID until the transaction completes, at which point either a SUCCESS or FAILURE message is returned to the requestor. The PENDING

status SHALL contain the value of TransactionID supplied in the request if one exists. If the requestor did not supply a transactionID, the server SHALL produce a TransactionID value and return this value in the PENDING response.

During the PENDING interval, the server may receive a query against the status of the transaction (see, for example, Section 5.7.4

GetCertInitial). If the identified transaction is still pending, the server SHALL respond with another PENDING response containing the previously stored value for TransactionID. If however no transaction is in a PENDING state that matches the TransactionID specified in a status query, the server SHALL respond with an empty PKIData envelope (i.e. no message body) containing the value of FAILURE for pkiStatus and BADTRANSID for failInfo.

4.7 Requirements on Data Origin Authenticity

All messages SHALL be digitally signed.

Prior to accepting a message as valid, clients, servers or services SHALL confirm that:

1. The signature on the request is valid; and
2. The certificate used to validate the signature is not revoked.

In the instance when the request is a certification request of a signature public key originating directly from an end-entity, the signature should not be relied upon as an assertion of authenticity of the attributes contained in the request. This authentication is function of the CAÂs role upon receipt of a certification request.

Prior to accepting a response as valid, clients SHALL confirm that:

1. The response corresponds to a former request; and
2. The identity of the response originator matches the intended recipient of the prior request.

4.8 Requirements on Use of CRMF and CMMF

The following table establishes implementation requirements on the implementation of message bodies as defined in [CRMF] and [CMMF]. Unless otherwise identified in the table, CMC implementations MAY implement additional CMMF messages to meet the needs of specific environments.

Message	SHALL	SHOULD
PKCSReq		X
CertReqMessages	X	
CertRep	X	
GetCert	X	
GetCertInitial	X	
GetCRL	X	
RevRequest	X	

The following sections establish additional requirements on the use of some message bodies.

4.8.1 PKCSReq

As defined in [CMMF], a PKCSReq message consists of a PKCS10 certificate signing request and additional registration information. The PKCS10 field SHOULD contain a ChallengePassword attribute generated by the owner of the private key. It MAY contain an optional ExtensionReq attribute used to indicate to the server one or more PKIXCERT certificate extensions. It MAY contain additional attributes as specified by PKCS9.

Some products are operated in a fashion that assigns subject names from a central repository of information upon receipt of a public key for certification. To accommodate this mode, the value of subject name in a PKCSReq MAY be zero length but the field MUST be present.

CMS requires that the signerInfo contain a issuerNameSerialNumber value;

however for this transaction, the certificate has yet to be issued and therefore the serialNumber has not yet been assigned. Thus the issuerName and SerialNumber value in the signerInfo element of PKCSReq SHALL be set to NULL and zero, respectively.

4.8.2 Use of RegInfo in PKCSReq

The RegInfo field of a PKCSReq request MAY contain additional information relevant to the request. This information is supplied external to the PKCS10 object and as such is not a component of the proof of possession calculation (see [CMMF]).

The RegInfo field is intended to consist of name-value pairs. Appendix B.1 of [CRMF] defines some standard name-value pairs and associated encoding mechanisms. Clients SHALL use those mechanisms when using the RegInfo field of PKCSReq to address the requirements met by the elements defined in CRMF Appendix B.1. Additional name-value pairs MAY be defined for environments with a need.

4.8.3 CertRep

A CertRep message is the response to a CertReqMessages, PKCSReq, GetCert or GetCRL. It is defined in [CMMF].

The following requirements pertain to the construction of CertRep message in response to receipt of PKCSReq message:

CertRep message consists of one or more certificates and an associated RspInfo field. The certificate(s) are contained in the response field. This field is a SignedData CMS content type with no ContentInfo or SignerInfos fields. Section 5.1 of CMS further describes this particular construction. In particular, although the SignedData construct is used, no signature is produced.

If the value of pkiStatus service indicator in a CertRep message is SUCCESS, the response field SHALL contain the requested certificate(s) and MAY contain additional non-root CA certificates related to the validation of the requested certificate(s).

If the value of pkiStatus service indicator in a CertRep message is PENDING or FALURE, the response field SHALL be excluded.

In all cases, the rspInfo field of CertRep MAY contain additional information useful to the application receiving the message in the form of name-value pairs. Other additional data could include detailed error information that uniquely identifies an information field supplied in the original regInfo field of a PKCSReq; e.g. the Zip Code was incorrect for the specified address or locale or billing failed because credit card information was in error.

CAs that receive a PKCSReq with a null subject name MAY reject such requests. If rejected, the CA SHALL respond with a CertRep message with pkiStatus of FAILURE and failInfo value of BADREQUEST.

The client MAY incorporate one or more standard X.509 v3 extensions in

the request as defined in [CMMF]. Servers are not required to be able to process every v3 X.509 extension transmitted using this protocol, nor are they required to be able to process other, private extensions. However, in the circumstance when a certification request is denied due to the inability to handle a requested extension, the server SHALL respond with a CertRep message indicating FAILURE and a corresponding FailureInfo field with the value of UNSUPPORTEDEXT.

4.8.4 GetCertInitial

The GetCertInitial message is used to poll for the certificate associated with prior PKCSReq request if the response to that prior request was a CertRep message with a status of PENDING.

Certificates are normally referenced using the CA DN and the certificate serial number. In this case however a certificate has not yet been issued. Thus the CA and Subject DNs are present in the message as a means to identify the requested certification. [CMMF] defines this syntax.

Clients that implement GetCertInitial SHALL conform to the requirements on use of the TransactionID service indicator as defined in Section 5.4.2.

4.8.5 GetCert

This operation is used to retrieve certificates from a certificate server or serviceÂs repository. This message is useful in circumstances where a fully-deployed Directory is either infeasible or undesired. Additionally, certificate retrieval requests may be used by resource-constrained clients to recover their public key in the event of a device reset. This could be the case within low-end IP routers that do not retain their certificates in non-volatile memory.

The GetCert message body consists of the CertID syntax defined in [CMMF]. For reference, this syntax is:

```
CertID ::= SEQUENCE {
    issuerName  GeneralName,
    serialNumber  INTEGER }
```

The response to a GetCert message SHALL consist of CertRep message containing a signedData response containing the requested certificate as defined in [CMMF].

4.8.6 GetCRL

This operation is used to retrieve CRLs from a CA's repository. In order to provide clients a convenient means of determining the network address needed to acquire a CA's CRL, servers and clients SHOULD be capable of producing and processing the CRLDistributionPoints certificate extension as defined in [PKIXCERT].

The response to a GetCRL message SHALL consist of CertRep message containing only a value for CRLs as a component of a ÂdegenerateÂ SignedData rather than the certificates component. Contents for the rspInfo field of CertRep MAY be included.

4.8.7 RevReq

[CMMF] defines the syntax of a RevReq (revocation request). For reference, a RevReq message body consists of:

- name of the certificate issuer
- serial number of the certificate to be revoked
- a reason code
- an optional passphrase
- an optional comment field

For a revocation request to become a reliable object in the event of a dispute, a strong proof of originator authenticity is required. A Registration AuthorityÂs digital signature on the request can provide this proof for certificates within the scope of the RAÂs revocation authority. The means by which an RA is delegated this authority is a matter of operational policy.

However, in the instance when an end-entity has lost use of their signature private key, it is impossible to produce a reliable digital signature. The PKCSReq message provides for the optional transmission from the CA to the end-entity of a passphrase which may be used as an alternative authenticator in the instance of loss of use. The acceptability of this practice is a matter of local security policy.

Clients SHALL provide the capability to produce a digitally signed RevReq message. Clients SHOULD provide the capability produce an unsigned revocation request containing the end-entityÂs passphrase. If a client provides passphrase-based self-revocation, the client SHALL be capable of producing a PKCSReq containing a passphrase.

The structure of an unsigned, passphrase-based RevReq is a matter of local implementation. Since such a message has no relationship to the use of cryptography, the use of CMS to convey this message is not required.

The response to a RevReq message SHALL consist of CertRep message with the following characteristics:

Success:
1. The pkiStatus service indicator SHALL contain a value of SUCCESS.
2. The response field of the CertRep message body MAY be included. If so, it SHALL contain the updated CRL relevant to the subject certificate.

Failure:
1. The pkiStatus service indicator SHALL contain a value of FAILURE.
2. The failInfo service indicator shall be set to the appropriate reason code.
3. The response field of the CertRep message body SHALL be excluded.

In either case, the rspInfo field of CertRep message body MAY be included. If so, it SHALL contain additional information relevant to the interpretation of the failure.

5. Security Considerations

Initiation of a secure communications channel between an end-entity and a CA necessarily requires an out-of-band trust initiation mechanism. For example, a secure channel may be constructed between the end-entity and the CA via IPSEC or TLS. Many such schemes exist and the choice of any particular scheme for trust initiation is outside the scope of this document. Implementors of this protocol are strongly encouraged to consider generally accepted principles of secure key management when integrating this capability within an overall security architecture.

Mechanisms for thwarting replay attacks may be required in particular implementations of this protocol depending on the operational environment. In cases where CAs maintain significant state information themselves, replay attacks may be detectable without the inclusion of the optional nonce mechanisms. Implementors of this protocol need to carefully consider environmental conditions before choosing whether or not to implement the senderNonce and recipientNonce service indicators described in section 4.3.1. Developers of state-constrained PKI clients are strongly encouraged to incorporate the use of these service indicators.

6. References

[CMS] R. Housley, "Cryptographic Message Syntax", draft-ietf-smime-cms-01.txt, October 1997

[PKCS7] B. Kaliski, "PKCS #7: Cryptographic Message Syntax v1.5", draft-hoffman-pkcs-crypt-msg-03.txt, October 1997

[PKCS10] B. Kaliski, "PKCS #10: Certification Request Syntax v1.5", draft-hoffman-pkcs-certif-req-03.txt, October 1997

[PKIXCERT] R. Housley, W. Ford, W. Polk, D. Solo "Internet Public Key Infrastructure X.509 Certificate and CRL Profile", draft-ietf-pkix-ipki-part1-09.txt, July, 1998

[CRMF] M. Myers, C. Adams, D. Solo, D. Kemp, Certificate Request Message Format, <draft-ietf-pkix-crmf-01.txt>, May, 1998

9. Author's Addresses

Michael Myers
VeriSign, Inc.
1390 Shorebird Way
Mountain View, CA, 94019
(650) 429-3402
mmyers@verisign.com

Xiaoyi Liu
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134

(480) 526-7430
xliu@cisco.com

Barbara Fox
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
(425) 936-9542
bfox@microsoft.com

Jeff Weinstein
Netscape Communications Corporation
501 Middlefield Road
Mountain View, CA 94043
jsw@netscape.com