SMIME Working Group
Internet Draft

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

expires in six months
November 1997

Certificate Request Syntax

<draft-ietf-smime-crs-00.txt>

1. Status of this Memo

2. Abstract

This document defines an Internet PKI Certificate Request Syntax (CRS). It addresses a growing need within the Internet PKI community for an interface to public key certification products and services based on PKCS7 [PKCS7] and PKCS10 [PKCS10]. A small number of additional services are defined to supplement the core certificate request service. Current industry practice regarding the use of PKCS7 and PKCS10 is also documented for the benefit of the Internet community.

In general, the use of PKCS7 in this document is aligned to the Cryptographic Message Syntax [CMS] which provides a superset of the PKCS7 syntax. Throughout this document, the term CMS should be taken to include the PKCS #7 document as defined in [PKCS7]. The term CRS refers to this specification.

The chief differences between CRS and PKIXMGMT are:

- Use of PKCS7 for security encapsulation and transaction framework
- Use of PKCS10 as the certification request message content
- Certification of Diffie-Hellman Public Keys based on PKCS10 requests
- No assumption of reliable connectivity or persistent on-line operation
- Single request/response transaction model

The key words "MUST", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119.

3. Protocol Overview

This document identifies use of some CMS protocol elements. In the interests of brevity and syntactic alignment, readers interested in the full syntactic context should refer to that document.

A transaction in this specification is composed of the exchange of request-response message pairs between a server and a client. Three transactions are defined:

1. **Certification of a public key;**
2. **Certificate and CRL retrieval; and**
3. **Certificate revocation.**

A public key certification request is formed as a PKCS10 object. The corresponding response is CMS encapsulation of the requested certificate and its associated non-Root CA certificate(s). Within this draft, the latter construction is referred to as a certificate response, or CertRep. A CertRep is a CMS signedData object that only contains certificates. Both the PKCS10 content and the resulting CertRep SHOULD be encapsulated within a full CMS security envelope. Encapsulating the CertRep in a full CMS envelope enables Registration Authorities to assert a signature across a certificate request. Some existing implementations of these mechanisms utilize a simplified form of this exchange. In these implementations, an unencapsulated PKCS10 object is provided to the Certification Authority, which responds with an unencapsulated CertRep syntax.

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.

Transactions MAY 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.

Replay protection MAY be 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.

Message bodies MAY be encrypted. Whether encrypted or cleartext, either are in turn included as the content of SignedData. The message digest of message body is then signed together with the Service Indicators using the originator's private signing key, producing the message signature.

The following figure illustrates the essential characteristics of fully-encapsulated transaction message structure. Many interior details and fields have been omitted for clarity. Not all of the indicated fields are present in every message type. The ORIGINATOR S CERTIFICATES element may actually be composed of one or more non-root CA certificate together with the end-entity certificate needed to validate the message signature.

```
|--------------------------------------|
|SIGNED DATA                           |
|--------------------------------------|
|    content                           |
|        DATA                          |
|           cleartext message body     |
|        or                            |
|        ENCRYPTED DATA                |
|           Message Encryption Key (MEK)| - encrypted MEK
|           encrypted message body     | - encrypted under MEK
|--------------------------------------|
|    ORIGINATOR'S CERTIFICATES         |
|--------------------------------------|
|    AUTHENTICATED ATTRIBUTES          |
|        Hash of Content               |
|        version                       | - protocol version number
|        message type                  | - indicates msg body syntax
|        transaction status            |
|        transaction identifier        | - optional
|        sender nonce                  | - optional
|        recipient nonce               | - optional
|--------------------------------------|
|    Message Signature                 | -produced by originator
|--------------------------------------|
```

4. Protocol Elements

4.1 PKCS7 Interoperability

CMS differs from PKCS7 in the following ways:

- 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 and MailListKeyIdentifier are optional within this specification.

As noted in CMS, if either RecipientKeyIdentifier or MailListKeyIdentifier are 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

CRS 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. PKI clients and servers SHOULD also be capable of producing and processing RSA signatures as specified in section 7.2.1 of PKIXCERT.

CRS 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. PKI 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.

Two options exist with respect to the use of encryption. One usage produces an encrypted message body encapsulated by a signed, cleartext envelope. Organizations that wish to protect against the leakage of sensitive data via cleartext channels may chose instead to produce a cleartext message body which is placed with a signed envelope which is in turn encapsulated by an encrypted envelope. The following figure illustrates the options:

```
Option 1                         Option 2
--------                         --------
SignedData                       EnvelopedData
  EnvelopedData                    SignedData
    Data                              <message body>
      <message body>
```

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

4.3 Message Construction

A fully-encapsulated CRS message SHALL be constructed as follows. The contentInfo field of a signedData type is populated with the envelopedData content type if the content is encrypted or Data content type if the data is sent in cleartext form (see the Open Issues section at the end of this document regarding MIME agent interoperability). The content MAY contain one of the following message bodies:

- PKCSReq -- Request for a certificate
- CertRep -- Response to a certificate request
- GetCert -- A means to query for others' certificates
- GetCRL -- A means to query for a CA's CRL(s)
- DualReq -- Separate signature and encryption certificates
- DualRep -- Responses to a DualReq
- RevReq -- Request to revoke a certificate
- RevResp -- Response to a revocation request

The presence of a particular message body type in either the encryptedContent of an envelopedData content type or the content of a Data content type SHALL be indicated by value of the messageType Service Indicator (see section 4.4) of the outermost SignedData. Successful responses are indicated by a value of SUCCESS for pkiStatus. All other responses are indicated by a value of FAILURE.

A PKCSReq message consists of a PKCS10 certificate signing request. The request SHALL contain subject name and subject public key. The subject name MAY be NULL, but must be present. The request MAY contain a ChallengePassword attribute. 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.

The ExtensionReq attribute is composed as a SEQUENCE OF X.509 extensions. When present in a PKCS10 certification request, this attribute SHALL be identified by the OID {pkcs-9 14} (This OID should be assiged off an SMIME WG arc?).

A CertRep message is the CA's response to a PKCSReq, GetCert or GetCRL. It MAY contain a certificate but always contains one or more Service Indicators.

The DualReq and DualRep message pair enables production and processing of separate signature and encryption certification request in a single message.

The RevReq and RevRep message pair enables production and processing of certificate revocation requests.

4.3.1 Service Indicators

The SignerInfos portion of SignedData carries one or more Service Indicators as authenticatedAttributes. As defined in CMS, the value of authenticatedAttributes is hashed using the algorithm specified by digestAlgorithm, signed using the originator's private key corresponding to digestEncryptionAlgorithm, the result encoded as an OCTET STRING and assigned to the encryptedDigest field of SignedData.

The following Service Indicators are defined:

- version
- messageType
- pkiStatus
- failinfo
- transactionId
- senderNonce
- recipientNonce

Each is uniquely identified by an Object Identifier (OID) that signals the syntax following the OID. Processing systems would first detect the OID and process the corresponding service indicator value prior to processing the message body.

As noted, service indicators are encoded as AuthenticatedAttributes within SignedData. In accordance with CMS, when included in a CMS message, AuthenticatedAttributes must consist of at a minimum:

- A PKCS #9 content-type attribute having as its value the content type of the ContentInfo value being signed.

- A PKCS #9 message-digest attribute, having as its value the message digest of the content.

CRS clients and servers SHALL include values for AuthenticatedAttributes in accordance. In addition to the required Attributes of content-type and message-digest, one or more of the following six authenticated attributes MUST be included as needed.

Service Indicator	OID	Syntax
version	smime-crs 1	INTEGER
TransactionId	smime-crs 2	INTEGER
messageType	smime-crs 3	INTEGER
pkiStatus	smime-crs 4	INTEGER
failInfo	smime-crs 5	INTEGER
senderNonce	smime-crs 6	OCTET STRING
recipientNonce	smime-crs 7	OCTET STRING
DualStatus	smime-crs 8	{as specified below}

```
DualStatus ::= SEQUENCE OF {
    request_id      INTEGER,
    status          pkiStatus,
    failure         failInfo OPTIONAL }
```

The DualStatus syntax supports certificate request messages that contain request for more than one certificate. That is, an end-entity submits a DualReq request and receives a DualRep response containing a DualStatus service indicator.

The version service indicator indicate CRS protocol version. If absent, a value of 0 SHALL be assumed. This draft specifies CRS version 1.

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

The messageType service indicator specifies the type of operation performed by the transaction. This attribute is required in all messages. The following values for messageType are defined:

Message	MessageType Value
PKCSReq	(19) -- PKCS#10 certificate request
CertRep	(3) -- Response to certificate request
GetCert	(21) -- Retrieve a certificate
GetCRL	(22) -- Retrieve a CRL
DualReq	(23) -- Dual certificate requests in a single msg
DualRep	(24) -- Response to a DualReq

The value given for messageType value is set in the messageType service indicator for messages of the indicated type. This service indicator SHALL be included in every message.

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

Response messages SHALL include transaction status information which is defined as pkiStatus service indicator:

Status	pkiStatus value
SUCCESS	(0) -- request successful
FAILURE	(2) -- request rejected

This pKIStatus service indicator is required for all PKI messages. The value given for pkiStatus value is set in the pkiStatus service indicator for status of the indicated type.

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 following failure reasons:

```
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.
```

Additional failure reasons MAY be defined for environments with a need.

The messageType, pkiStatus, and failInfo service indicators are mandatory for all messages. 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 included the transmitted value.

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 include only a value for senderNonce. If included, 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. The recipient of a sender nonce reflects this value back to the originator as a recipientNonce and includes it's own senderNonce. Upon receipt by the transaction originator of this message, 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.

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.

If a transaction originator includes a value for the transaction-id service indicator in a service request, responses SHALL include this value as a value for transaction-id service indicator.

4.3.2 Messages Bodies

The following message bodies SHALL be constructed in the method indicated. Some current implementations transmit a PKCS10 request and its corresponding response directly. These quantities SHOULD be CMS-encapsulated prior to transmission as described above.

4.3.3 PKCSReq Message Body

A PKCSReq message body is composed of two elements. The first consists of the PKCS10 structure itself. The second consists of optional supplementary registration information. Syntactically, this message body is constructed as:

```
PKCSReq ::= SEQUENCE {
    csr         CertificationRequest,
    regInfo     OCTET STRING OPTIONAL }
```

```
CertificationRequest ::=         SEQUENCE {
  certificationRequestInfo         SEQUENCE {
    version                          INTEGER,
    subject                          Name,
    subjectPublicKeyInfo             SEQUENCE {
      algorithm                        AlgorithmIdentifier,
      subjectPublicKey                 BIT STRING }
    attributes                     [0] IMPLICIT SET OF Attribute }
  signatureAlgorithm               AlgorithmIdentifier,
  signature                        BIT STRING }
```

Clients SHALL produce a PKCS10 message body containing a subject name and public key. Some certification 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 of operation, the subject name in a PKCSReq MAY be NULL, but MUST be present. CAs that receive a request 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 an ExtensionReq attribute. An ExtensionReq attribute is defined as

```
ExtensionReq ::= SEQUENCE OF Extension
```

where Extension is imported from PKIX Part 1 and ExtensionReq is identified by {pkcs-9 14} (OID should be off the SMIME WG arc?).

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 an UNSUPPORTEDEXT error as defined in section 4.3.1.

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 SHALL be set to NULL and zero, respectively.

Schematically, the relationship among the elements of a PKCSReq message are as follows:

```
|---------------------------------------|
|SIGNED DATA                            |
|---------------------------------------|
|    DATA                               |
|        PKCSReq                        | - the PKCS10
|or  ENVELOPED DATA                     |
|        MEK                            | - encrypted key exchange key
|        encrypted content:             | - encrypted under MEK
|           PKCSReq                     | - the PKCS10
|---------------------------------------|
|    ORIGINATOR S CERTIFICATES          |
|---------------------------------------|
|    AUTHENTICATED ATTRIBUTES           |
|        Hash of Content                |
|        version                        | - protocol version number
|        message type                   | - PKCSREQ
|        transaction status             |
|        transaction identifier         | - optional
|        sender nonce                   | - optional
|---------------------------------------|
|    Message Signature                  |
|---------------------------------------|
```

4.3.4 Production of Diffie-Hellman Public Key Certification Requests

When a PKCSReq or DualReq message is used to produce a certified Diffie-Hellman public key, the means to establish proof of possession of the corresponding private key SHALL be performed using the methods described in Appendix A of this document.

CAs that support this mechanism SHALL have produced and make available a D-H public key certificate. This certificate SHALL contain the parameters associated with the public key.

Clients that support this mechanism SHALL generate a D-H key pair using the parameters contained in the CA s certificate they wish to have certify their D-H public key.

4.3.5 CertRep

A CertRep message is the response to a prior PKCSReq message. It is indicated by a value of CERTREP in the messageType service indicator. Successful responses are indicated by a value of SUCCESS for pkiStatus. All other responses are indicated by a value of FAILURE. Section 4.2 defines specific values for these constants.

4.3.5.1 FAILURE

Servers transmit a CertRep messages with a pkiStatus of FAILURE if the certification request or its subsequent processing fails to meet the certification practices requirements of the entity operating the server.

The message is a SignedData content type with no ContentInfo block. The SignerInfo block contains the following Authenticated Attributes as indicated:

```
|---------------------------------------|
|SIGNED DATA                            |
|---------------------------------------|
|---------------------------------------|
|    ORIGINATOR S CERTIFICATES          |
|---------------------------------------|
|    AUTHENTICATED ATTRIBUTES           |
|        Hash of Content                |
|        version                        | - protocol version number
|        message type                   | - CERTREP
|        transaction status             | - FAILURE
|        failInfo                       | - reason code
|        transaction identifier         | - from prior msg, if present
|        sender nonce                   | - prior msg, if present
|        recipient nonce                | - current server nonce if
|                                       |   prior client nonce present
|---------------------------------------|
|    Message Signature                  |
|---------------------------------------|
```

Failure reason codes are specified in Section 4.2.

4.3.5.2 SUCCESS response

The Client's certificate(s) SHALL be conveyed to the Client as a certs-only CMS SignedData message. To construct such a message, the certificates are made available to the CMS generating process which creates a CMS object of type signedData. The contentInfo and signerInfos fields must be empty. The message body SHALL include all non-root certificates needed to validate the signature on the end-entity's certificate(s).

Schematically, the general relationship among these elements is as follows:

```
|----------------------------------------|
|SIGNED DATA                             |
|----------------------------------------|
|    DATA                                |
|       CA certificate(s)                | - intermediate CA cert(s)
|       ee certificate                   | - end entity sig. certificate
|       ee certificate                   | - end entity enc. certificate
|or  ENVELOPED DATA                      |
|        MEK                             | - encrypted key exchange key
|        encrypted content:              | - encrypted under MEK
|           CA certificate(s)            | - intermediate CA cert(s)
|           ee certificate               | - end entity sig. certificate
|           ee certificate               | - end-entity enc. certificate
|----------------------------------------|
|    ORIGINATOR S CERTIFICATES           |
|----------------------------------------|
|    AUTHENTICATED ATTRIBUTES            |
|        Hash of Content                 |
|        version                         | - protocol version number
|        message type                    | - CERTREP
|        transaction status              | - SUCCESS
|        transaction identifier          | - from prior msg, if present
|        sender nonce                    | - prior msg, if present
|        recipient nonce                 | - current server nonce if
|                                        |   prior client nonce present
|----------------------------------------|
|    Message Signature                   |
|----------------------------------------|
```

This schematic depicts the situation where an end-entity has requested separate signature and encryption certificates in a single message. Section 4.3.6 which follows specifies the means by which an end-entity can perform a single request that yields this result.

4.3.6 DualReq and DualRep

This construct is used by a single end-entity to request signature and encryption certificates in a single message. For each certification, a request identifier is included.

This identifier is needed to differentiate among the multiple CertRep data that will be returned in the resulting DualRep response. The optional transaction identifier and sender nonce, if included, apply to the message as a whole.

Syntactically,

```
DualReq ::= SEQUENCE OF {
     request    PKCSReq,
     req_id     INTEGER }
```

The values for req_id SHALL be relatively unique within a single DualReq content.

A PKCSReq for key-usage differentiated certificates SHALL contain a value for KeyUsage extension identified and structured in accordance with PKIXCERT. This extension SHALL be conveyed as an ExtensionReq Attribute within the PKCS10 syntax.

For end-entities requesting the production of a signature-only certificate, the keyUsage extension SHALL contain either a setting of DigitalSignature, NonRepudiation or both.

For end-entities requesting the production of an encryption-only certificate, the keyUsage extension SHALL contain a setting of keyEncipherment, dataEncipherment, or keyAgreement.

Schematically, the resulting message would appear as follows:

```
|---------------------------------------|
|SIGNED DATA                            |
|---------------------------------------|
|    DATA                               |
|        PKCSReq, 1                     | - first  PKCS10
|        PKCSReq, 2                     | - second PKCS10
|or  ENVELOPED DATA                     |
|        MEK                            | - encrypted key exchange key
|        encrypted content:             | - encrypted under MEK
|          PKCSReq, 1                   | - first  PKCS10
|          PKCSReq, 2                   | - second PKCS10
|---------------------------------------|
|    ORIGINATOR S CERTIFICATES          |
|---------------------------------------|
|    AUTHENTICATED ATTRIBUTES           |
|        Hash of Content                |
|        version                        | - protocol version number
|        message type                   | - DUALREQ
|        transaction status             |
|        transaction identifier         | - optional
|        sender nonce                   | - optional
|---------------------------------------|
|    Message Signature                  |
|---------------------------------------|
```

The corresponding response to a DualReq message is composed of a similarly iterated construction:

```
DualRep ::= SEQUENCE OF {
      req_id     INTEGER,
      ee_certs   SEQUENCE OF CertRep }
```

4.3.7 GetCRL

This operation is used to retrieve CRLs from a CA's repository. It assumes the target CA maintains one and only one CRL relative to a given private signing key. 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. Alternatively, the Client MAY be pre-configured with the CRL's location.

```
|---------------------------------------|
|SIGNED DATA                            |
|---------------------------------------|
|    DATA                               |
|        GetCRL message body            |
|---------------------------------------|
|    ORIGINATOR S CERTIFICATES          |
|---------------------------------------|
|    AUTHENTICATED ATTRIBUTES           |
|        Hash of Content                |
|        version                        | - protocol version number
|        message type                   | - GETCRL
|        transaction status             |
|        transaction identifier         | - optional
|        sender nonce                   | - optional
|---------------------------------------|
|    Message Signature                  |
|---------------------------------------|
```

The GetCRL message body SHALL consist of the following syntax:

```
GetCRL ::= SEQUENCE {
    issuerName  Name,
    time          GeneralizedTime }
```

The Name component is the value of the Issuer DN in the subject certificate. The time component is used by the client to specify from among several issues of CRL that one whose thisUpdate value is less than but nearest to the specified time.

4.3.8 GetCert

A certificate retrieval request MAY be used by resource-constrained clients to recover their public key in the event of a device reset. This MAY be the case, for example, within low-end IP routers which may not be capable of retaining their certificates in non-volatile memory.

```
|----------------------------------------|
|SIGNED DATA                             |
|----------------------------------------|
|    DATA                                |
|        GetCert message body            |
|----------------------------------------|
|    ORIGINATOR S CERTIFICATES           |
|----------------------------------------|
|    AUTHENTICATED ATTRIBUTES            |
|        Hash of Content                 |
|        version                         | - protocol version number
|        message type                    | - GETCERT
|        transaction status              |
|        transaction identifier          | - optional
|        sender nonce                    | - optional
|----------------------------------------|
|    Message Signature                   |
|----------------------------------------|
```

The GetCert message body SHALL consist of the following syntax:

```
GetCert ::= SEQUENCE {
    issuerName  Name,
    serialNumber  INTEGER }
```

4.3.9 Revocation Request and Response

A revocation request is structured as follows.

```
RevReq ::= SEQUENCE {
    issuerName  Name,
    serialNumber  INTEGER,
    reason        ReasonFlags,             -- from PKIXCERT
    passphrase    OCTET STRING OPTIONAL,
    comment       PrintableString OPTIONAL }
```

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 Authority s digital signature on the request can provide this proof for certificates within the scope of the RA s 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.

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

A cleartext signed revocation request has the following basic structure:

```
|----------------------------------------|
|SIGNED DATA                             |
|----------------------------------------|
|    DATA                                |
|        RevReq message body             |
|----------------------------------------|
|    ORIGINATOR S CERTIFICATES           |
|----------------------------------------|
|    AUTHENTICATED ATTRIBUTES            |
|        Hash of Content                 |
|        version                         | - protocol version number
|        message type                    | - REVREQ
|        transaction status              |
|        transaction identifier          | - optional
|        sender nonce                    | - optional
|----------------------------------------|
|    Message Signature                   |
|----------------------------------------|
```

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 revocation request, REVRESP, is a CRS message with no message body. The services indicators will convey the status of the processing of the message by the recipient CA service or server. Its basic structure is as follows:

```
|----------------------------------------|
|SIGNED DATA                             |
|----------------------------------------|
|----------------------------------------|
|    ORIGINATOR S CERTIFICATES           |
|----------------------------------------|
|    AUTHENTICATED ATTRIBUTES            |
|        Hash of Content                 |
|        version                         | - protocol version number
|        message type                    | - REVRESP
|        transaction status              | - SUCCESS or FAILURE
|        failure                         | - present if status is FAILURE
|        transaction identifier          | - optional
|        sender nonce                    | - optional
|----------------------------------------|
|    Message Signature                |
|-------------------------------------|
```

6. 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.

7. Open Issues

This draft specifies the essential characteristics of a protocol based on the CMS and PKCS7 security encapsulation syntaxes. Integration of this protocol to MIME needs to be performed to align it with generally accepted practices of MIME interoperability. In particular it is noted that current SMIME draft specifications integrate CMS syntax with MIME encapsulation to achieve SMIME interoperability.

This draft also presents a number of optional constructions. Consensus needs to be established on a minimum interoperable profile.

8. References

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

[DH] B. Kaliski, "PKCS 3: Diffie-Hellman Key Agreement v1.4"

[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-part1-06.txt, October, 1997

[PKIXMGMT] C. Adams, S. Farrell, "Internet Public Key Infrastructure Certificate Management Protocols", draft-ietf-pkix-ipki3cmp-04.txt, September 1997

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

Hemma Prafullchandra
Sun Microsystems Inc
2550 Garcia Ave
CUP01-102
Mountain View, CA 94043
(408) 343-1798
hemma@sun.com

Appendix A

A PROPOSAL FOR CERTIFICATION REQUEST OF DIFFIE-HELLMAN PUBLIC VALUES

==

Version 1.2
September 26, 1996

1.0 INTRODUCTION

PKCS #10 [PKCS10]from RSA Labs defines a syntax for certification requests. In there it is assumed that the public key being requested for certification corresponds to an algorithm which is capable of signing/encrypting. Diffie-Hellman (DH) is a key agreement algorithm and thus cannot be used for signing/encrypting. Hence, the following enhancement to PKCS #10 is proposed for requesting certification for Diffie-Hellman public values.

2.0 SCOPE

This proposal is primarily concerned with "signatureAlgorithm" and "signature". I believe there has been some discussions on extending/revising PKCS #10 to allow for such things as validity period and optional fields supported in X.509 v3 certificates, none of these are addressed in this proposal.

Also this proposal does not provide for identity authentication. However this can be achieved if the enrollee already possesses a valid signature key certificate and the corresponding private key to this is used to sign the certification request (as opposed the private corresponding to the public key in the certification request itself).

3.0 DEFINITIONS

For the purposes of this proposal, the following definitions apply.

```
DH certificate = a certificate whose SubjectPublicKey is a DH public
                 value and is signed with any signature algorithm
                 (e.g. rsa or dsa).
```

4.0 SPECIFICATION FOR DH CERTIFICATION

>From PKCS #10, section 6.2 CertificationRequest, the following ASN.1 syntax is defined.

```
CertificationRequest ::= SEQUENCE {
        certificationRequestInfo       CertificationRequestInfo,
        signatureAlgorithm             SignatureAlgorithmIdentifier,
        signature                      Signature
}

SignatureAlgorithmIdentifier ::= AlgorithmIdentifier

Signature ::=   BIT STRING
```

4.1 DH CERTIFICATION PROCESS

The steps then for requesting certification of DH public values are:

1. An entity chooses a Certification Authority. It obtains the CAs' DH certificate. Other than trust, the selection has to take into account with which community/group of entities this entity wishes to use the certified DH public value (for interoperability a common set of DH parameters have to be used).

   [Note: a CA MAY have more than one signature key, and thus is likely to have more than one certificate. This proposal requires that a CA which wishes to provide certification service for public key agreement algorithms makes available a certificate with a public key (and common parameters) for that algorithm.]

   If the enrollee already possess a valid signature key certificate, then, the enrollee can select the group parameters g and P, and compute a DH public value, all this would be encoded as the SubjectPublicKeyInfo and the request signed by the signature key. The CA can then verify the request (it would need the enrollees signature key certificate to obtain the public key) and issue a certificate.

2. The entity generates a DH public/private key-pair.

   The DH parameters used to calculate the public should be those specified in the CAs' DH certificate.

```
From CAs' DH certificate:
     CApub = g^x mod p      (where g and p are the established DH
                              parameters and x is the CAs' private
                              DH component)
For entity E:
     DH private value = y
     Epub = DH public value = g^y mod p
```

3. The entity selects a SignatureAlgorithmIdentifier. A number of these may be defined, the following is recommended.

```
craWithHMACandSHA1 ::= { an oid needs to be assigned }
```

[certification request algorithm with HMAC-SHA1 as the message digest algorithm as specified in [HMAC-MD5].]

4. The signature process will then consist of:

a) The value of the certificationRequestInfo field is DER encoded, yielding an octet string (as per PKCS #10). This will be the `text' referred to in [HMAC-MD5], the data to which HMAC-SHA1 is applied.

b) A shared DH secret is computed, as follows,

shared secret = Kec = g^xy mod p

[This is done by the entity E as g^(y.CApub) and by the CA as g^(x.Epub), where CApub is retrieved from the CA's DH certificate and Epub is retrieved from the actual certification request.]

c) A key K is derived from the shared secret Kec as follows:

K = SHA1(DER-encoded-subjectName | Kec | DER-encoded-issuerName)

where "|" means concatenation.

d) Compute HMAC-SHA1 over the data `text' as per [HMAC-MD5] as:

SHA1(K XOR opad, SHA1(K XOR ipad, text))

where,

opad (outer pad) = the byte 0x36 repeated 64 times

and

ipad (inner pad) = the byte 0x5C repeated 64 times.

Namely,

(1) Append zeros to the end of K to create a 64 byte string (e.g., if K is of length 16 bytes it will be appended with 48 zero bytes 0x00).
(2) XOR (bitwise exclusive-OR) the 64 byte string computed in step (1) with ipad.
(3) Append the data stream `text' to the 64 byte string resulting from step (2).
(4) Apply SHA1 to the stream generated in step (3).
(5) XOR (bitwise exclusive-OR) the 64 byte string computed in step (1) with opad.
(6) Append the SHA1 result from step (4) to the 64 byte string resulting from step (5).
(7) Apply SHA1 to the stream generated in step (6) and output the result.

Sample code is also provided in [HMAC-MD5].

e) The output of (d) is encoded as a BIT STRING (the Signature).

5. The signature verification process requires the CA to carry out steps (a) through (d) and then simply compare the result of step (d) with what it received as the signature component. If they match then the following can be concluded:

   1) The Entity possesses the private key corresponding to the public key in the certification request because it needed the private key to calculate the shared secret; and
   2) That only the CA, the entity sent the request to could actually verify the request because the CA would require its own private key to compute the same shared secret. This protects from rogue CAs.

5.0 ACKNOWLEDGEMENTS

I would like to thank Peter Williams and Dave Solo for providing comments which resulted in making this proposal clear, succinct and complete. Also, Hugo Krawczyk, John Kennedy, Bob Jueneman and Carlisle Adams provided extremely important improvements.

6.0 AUTHOR INFORMATION

Hemma Prafullchandra
Sun Microsystems Inc
2550 Garcia Ave
CUP01-102
Mountain View
CA 94043

(408) 343-1798
hemma@sun.com

7.0 REFERENCES

[PCKS10]	RSA Data Security, Inc., Public-Key Cryptography Standards (PKCS) #10: Certification Request Syntax Standard, November 1, 1993
[RFC1423]	D. Balenson, Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers, February, 1993
[HMAC-MD5]	H. Krawczyk, M. Bellare, R. Canetti, HMAC-MD5: Keyed-MD5 for Message Authentication, <draft-ietf-ipsec-hmac-md5-00.txt>, March, 1996