Network Working Group
Request for Comments: 4212
Category: Informational

M. Blinov
Guardeonic Solutions
C. Adams
University of Ottawa

October 2005

Alternative Certificate Formats for the Public-Key Infrastructure Using X.509 (PKIX) Certificate Management Protocols

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

This document is not a candidate for any level of Internet Standard. The IETF disclaims any knowledge of the fitness of this document for any purpose, and in particular notes that it has not had IETF review for such things as security, congestion control, or inappropriate interaction with deployed protocols. The RFC Editor has chosen to publish this document at its discretion. Readers of this document should exercise caution in evaluating its value for implementation and deployment.

Abstract

The Public-Key Infrastructure using X.509 (PKIX) Working Group of the Internet Engineering Task Force (IETF) has defined a number of certificate management protocols. These protocols are primarily focused on X.509v3 public-key certificates. However, it is sometimes desirable to manage certificates in alternative formats as well. This document specifies how such certificates may be requested using the Certificate Request Message Format (CRMF) syntax that is used by several different protocols. It also explains how alternative certificate formats may be incorporated into such popular protocols as PKIX Certificate Management Protocol (PKIX-CMP) and Certificate Management Messages over CMS (CMC).

1. Introduction

Full certificate life-cycle management in a Public-Key Infrastructure (PKI) requires protocol support in order to achieve automated processing and end user transparency. Such protocols require standardization in order to allow more than one vendor to supply various pieces -- End Entity (EE), Certification Authority (CA), Registration Authority (RA) -- in the PKI deployment of a single organization, or to allow multiple, independently-deployed PKIs to be interconnected usefully.

The IETF PKIX (Public-Key Infrastructure using X.509) Working Group currently has several certificate management protocols and certificate request syntax specifications on the standards track. Although these specifications are primarily focused on X.509v3 public-key certificates, some of them can be easily extended to handle certificates in alternative formats as well.

This document focuses on a popular certificate request syntax called CRMF (Certificate Request Message Format) [CRMF]. Although the original specification of CRMF is X.509-specific, extensions have already been proposed to allow for alternative certificate templates [CMP]. However, those extensions have only defined a framework; they did not define the exact format to be used for various certificate types.

This document builds on top of the framework mentioned above and defines how CRMF can be used to request certificates of the following types:

- X.509 attribute certificates [ATTCERT]

- OpenPGP certificates [OPENPGP]

The CRMF syntax is used by such popular protocols as PKIX-CMP (PKIX Certificate Management Protocol) [CMP] and CMC (Certificate Management Messages over CMS) [CMC]. This means that CRMF extensions proposed in this document enable these protocols to request certificates of the above types. However, it is not enough to be able to request a certificate. The protocol should be prepared to handle certificates of a particular type and, for example, return them to the user.

This document proposes extensions to the PKIX-CMP and CMC protocols that are required to manage certificates in alternative formats.

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 RFC 2119 [RFC2119].

2. Certificate Template

One of the features of the CRMF format is its use of the CertTemplate construct, which allows a requester (EE, or RA acting on behalf of an EE) to specify as much or as little as they wish regarding the content of the requested certificate. It is explicitly noted that the CA has final authority over the actual certificate content; that is, the CA may alter certificate fields or may add, delete, or alter extensions according to its operating policy (if the resulting certificate is unacceptable to the EE or RA, then that certificate may be rejected and/or revoked prior to any publication/use).

A similar flexibility in the request must be available for alternative certificate types as well. For this purpose, an AltCertTemplate extension was introduced in [CMP] as follows (where id-regCtrl = {1 3 6 1 5 5 7 5 1}, as defined in [CRMF]).

```
CertRequest ::= SEQUENCE {
    certReqId     INTEGER,
    certTemplate  CertTemplate,
    controls      Controls OPTIONAL }

-- If certTemplate is an empty SEQUENCE (i.e., all fields
-- omitted), then controls MAY contain the
-- id-regCtrl-altCertTemplate control, specifying a template
-- for a certificate other than an X.509v3 public-key
-- certificate.  Conversely, if certTemplate is not empty
-- (i.e., at least one field is present), then controls
-- MUST NOT contain id-regCtrl-altCertTemplate.  The new
-- control is defined as follows:
id-regCtrl-altCertTemplate OBJECT IDENTIFIER ::= {id-regCtrl 7}
AltCertTemplate ::= AttributeTypeAndValue
```

In this section, an AltCertTemplate is specified for each of the alternative certificate types defined in Section 1.

2.1. X.509 Attribute Certificate CertTemplate

A CertTemplate for an X.509 attribute certificate can be used by simply defining an object identifier (OID) and corresponding value for use in the id-regCtrl-altCertTemplate control. These are specified as follows.

OID:

```
   id-acTemplate OBJECT IDENTIFIER ::=
      {id-regCtrl-altCertTemplate 1}
```

Value:

```
   AttCertTemplate ::= SEQUENCE {
      version                 AttCertVersion            OPTIONAL,
      holder                  Holder                    OPTIONAL,
      issuer                  AttCertIssuer             OPTIONAL,
      signature               AlgorithmIdentifier       OPTIONAL,
      serialNumber            CertificateSerialNumber   OPTIONAL,
      attrCertValidityPeriod  OptionalAttCertValidity   OPTIONAL,
      attributes              SEQUENCE OF Attribute     OPTIONAL,
      issuerUniqueID          UniqueIdentifier          OPTIONAL,
      extensions              Extensions                OPTIONAL
   }
   OptionalAttCertValidity  ::= SEQUENCE {
      notBeforeTime  GeneralizedTime  OPTIONAL,
      notAfterTime   GeneralizedTime  OPTIONAL
   } -- at least one must be present
```

2.2. OpenPGP Certificate CertTemplate

Similar to certificate templates defined above, a CertTemplate for an OpenPGP certificate can be used by defining an object identifier (OID) and corresponding value for use in the id-regCtrl-altCertTemplate control. These are specified as follows:

OID:

```
   id-openPGPCertTemplateExt OBJECT IDENTIFIER ::=
      {id-regCtrl-altCertTemplate 2}
```

Value:

```
   OpenPGPCertTemplateExtended ::= SEQUENCE {
      nativeTemplate   OpenPGPCertTemplate,
      controls         Controls  OPTIONAL }

   OpenPGPCertTemplate ::= OCTET STRING
   -- contains the OpenPGP CertTemplate data structure defined
   -- below (binary format without Radix-64 conversions)
   -- encoded as an ASN.1 OCTET STRING
```

2.2.1. OpenPGP CertTemplate Data Structure

Similar to the X.509 CertTemplate, the OpenPGP CertTemplate is an OpenPGP certificate (OpenPGP Transferable Public Key) [OPENPGP] with all fields optional. The essential elements of an OpenPGP CertTemplate are:

- Zero or one Public Key packet.

- Zero or more Direct Key Self Signature packets.

- Zero or more Certification Signature packets (only if no User ID packets are present).

- Zero or more User ID packets.

- After each User ID packet, zero or more Certification Signature packets.

- Zero or more Subkey packets.

- After each Subkey packet, zero or one Subkey Binding Signature packet.

Each packet in the OpenPGP CertTemplate MUST be a syntactically correct OpenPGP packet. This will enable conformant implementations to use existing PGP libraries for building and parsing OpenPGP CertTemplates.

The following implications of this rule should be explicitly noted:

- Fields for which the OpenPGP specification defines a set of permitted values (e.g., the signature type or the public key algorithm fields of the Signature packet) MUST have a value from the defined set. Even if the requester does not have any particular preferences for, for example, the signature algorithm, it MUST choose one value that is the most desirable.

  Rationale: An alternative solution could be to define extra "any" values, but this would be a modification of the OpenPGP syntax, which is not considered appropriate in this document.

- All subpackets of the Signature packet defined by the OpenPGP specification as mandatory (e.g., the creation time and the issuer’s key id subpackets) MUST be present even though they do not make much sense in the context of a certificate request.

- The number of MPIs at the end of the Key Material and the Signature packets MUST match the number defined by the OpenPGP specification for the given algorithm (the algorithm is controlled by the value of the "algorithm" field). For example, there should be 2 MPIs for DSA signatures. Note that the OpenPGP specification does not define validation rules for the content of those MPIs.

Though it is not considered appropriate here to define extra "any" values for fields of enumerated types, such values can still be defined for some other fields where the OpenPGP specification is not that strict.

The following extra values are defined in the context of the OpenPGP CertTemplate. Note that these definitions do not modify the syntax of OpenPGP packets, and existing PGP libraries can still be used to generate and parse them.

- For fields representing time (e.g., signature creation time): the value of zero means "any time".

- For fields holding key IDs: the value of 0xFFFFFFFFFFFFFFFF means "any key id".

- For signature fields: the "any signature" value is encoded as a sequence of MPIs such that:

  * the number of MPIs matches the number of MPIs defined by the OpenPGP specification for the given algorithm, and

  * the value of each MPI is 0xFF.

  A Signature packet with the "any" value in the signature fields is called a Signature Template.

    Example: The "any signature" value for a DSA signature would look like [00 08 FF 00 08 FF]

- For key material fields: the "any key" value is encoded as a sequence of MPIs such that:

  * the number of MPIs matches the number of MPIs defined by the OpenPGP specification for the given algorithm, and

  * the value of at least one of the MPIs is a bit string with all its bits set to 1.

A Key Material packet with the "any" value in the key material fields is called a Key Template. (See Key Template section for further details.)

Example: The "any key" value for a DSA public key may look like [00 08 FF 00 10 FF FF 00 10 85 34 00 08 FF]

The following rules apply to the sequence of packets within the OpenPGP CertTemplate:

- If the Public Key packet is omitted from the OpenPGP CertTemplate, then this CertTemplate does not constrain the value of the public key (i.e., it refers to "any" public key).

- The order of Signature packets following a User ID packet and the order of User ID packets within the CertTemplate are not important.

- If an OpenPGP CertTemplate does not contain any User ID packets, then it refers to "any" user IDs that are relevant to a given request.

2.2.2. OpenPGP CertTemplate in Certificate Requests

Since an OpenPGP certificate can have several certification signatures, the OpenPGP CertTemplate uses Signature Templates to define where certification signatures should occur. The values of the fields of the Signature Templates define the parameters of the new certification signatures. The following rules apply:

- A Signature Template that is present in the list of signatures following a User ID packet requests that the CA certify this User ID and the public key and replace the Signature Template with the new certification signature. The Signature Template does not mandate the exact place of the certification signature within the list. The certification signature may be inserted at any position within the list of signatures (following the certified User ID packet).

- A Signature Template may be present in the OpenPGP CertTemplate without any preceding User ID packet. In this case, it is assumed that the CA knows the ID(s) of the user by some other means. A Signature Template without a preceding User ID requests that the CA insert all known User IDs of the user into the OpenPGP certificate and certify each of them. The Signature Template defines the parameters of these certification signatures.

- If an OpenPGP CertTemplate contains no Signature Templates, then the CA is requested to certify all User IDs that are present in the OpenPGP CertTemplate. Such a CertTemplate does not define parameters of the certification signatures explicitly, but the CA SHOULD use parameters of the certification self-signatures (if present in the CertTemplate) as a guide (e.g., key flags fields).

- If neither Signature Templates nor User IDs are present in the OpenPGP CertTemplate, then the CA is expected to know the ID(s) of the user by some other means. In this case, the CertTemplate requests that the CA insert these User IDs into the OpenPGP certificate and certify each of them. The parameters of the certification signatures are left to the CA.

If several certification signatures have to be produced according to an OpenPGP CertTemplate, and any of them cannot be granted (even with modifications) for whatever reason, then the whole request with this OpenPGP CertTemplate MUST be rejected.

The client SHOULD provide enough information in its request that the CA could produce a complete OpenPGP certificate. For example, the client SHOULD include in the template all relevant subkeys with their binding signatures so that the CA can include them in the resultant OpenPGP certificate as well. Rationale: In some environments, the CA/RA is responsible for publishing certificates.

2.2.3. Key Templates and Central Key Generation

The OpenPGP CertTemplate can also be used to request certification of centrally-generated keys. This is accomplished by using Key Templates.

If the Public Key packet of an OpenPGP CertTemplate is a Key Template, then this OpenPGP CertTemplate requests that the CA/RA generate the key pair prior to certifying it. Fields of the Key Template define parameters of the new key pair as follows (see examples in the Appendix):

- The "public key algorithm" field specifies the algorithm to be used for the key generation.

- MPI fields with the value of 0xFF ([00 08 FF]) specify that no constraint is placed on the corresponding part of the key.

- MPI fields that contain any other bit strings in which all bits are set to 1, specify that the corresponding part of the key should be of the same length as the length of the MPI (e.g., the length of the public modulus n of the RSA key).

- MPI fields that contain any other values specify that the corresponding part of the key should be of the given value (key generation parameters).

In order to return a complete OpenPGP certificate, in addition to certifying the new key and the User ID, the CA (or RA) SHOULD also create a self-signature (i.e., sign the new public key and the User ID with the new private key) and include it after the User ID packet. This SHOULD be done for all User IDs certified by the CA.

If a Subkey packet of an OpenPGP CertTemplate is a Key Template, then this OpenPGP CertTemplate requests that the CA/RA generate a subkey. Fields of the Key Template define parameters of the new subkey. The new subkey obviously does not have to be certified. However, the CA/RA SHOULD produce the binding signature and include it after the subkey, if the CA/RA knows the user’s primary private key (e.g., it was centrally generated as well). Note that if the CA/RA does not know the user’s primary private key, then the resultant OpenPGP certificate returned from the CA/RA to the client will be incomplete (i.e., there will be no binding signature for the subkey). It will be the responsibility of the client to produce and add the binding signature and to publish the final OpenPGP certificate.

If an OpenPGP CertTemplate contains neither PublicKey/Subkey packets nor Key Template packets, then it requests that the CA generate keys/subkeys according to the CA’s policies.

2.2.4. OpenPGPCertTemplateExtended

The OpenPGPCertTemplateExtended structure enables additional extensions and controls to be added to the basic OpenPGP CertTemplate.

2.2.5. OpenPGP CertTemplate Required Profile

A conformant implementation is REQUIRED to support OpenPGP CertTemplates that are valid OpenPGP certificates, i.e., that have the following structure (see examples in the Appendix):

- One Public Key packet (not a Key Template).

- Zero or more Direct Key Self Signature packets (without Signature Templates).

- One or more User ID packets.

- After each User ID packet, zero or more Certification Signature packets (without Signature Templates).

- Zero or more Subkey packets (without Key Templates).

- After each Subkey packet, one Subkey Binding Signature packet (not a Signature Template).

A conformant implementation is REQUIRED to recognise Key Templates and Signature Templates and is REQUIRED to either support them or reject requests containing them if it does not.

3. Proof-of-Possession

A CRMF request includes a Proof-of-Possession (POP) field that contains proof that an End Entity has possession of the private key corresponding to the public key for which a certificate is requested.

The following rule applies to this field (with modifications from [CMP]):

```
* NOTE: If CertReqMsg certReq certTemplate (or the
* altCertTemplate control) contains the subject and
* publicKey values, then poposkInput MUST be omitted
* and the signature MUST be computed on the DER-encoded
* value of CertReqMsg certReq (or the DER-encoded value
* of AltCertTemplate).
```

An OpenPGP CertTemplate is considered to satisfy the conditions of this note if it has a Public Key packet (not a Key Template) and at least one User ID packet.

4. Protocol-specific Issues

This section explains how alternative certificate formats may be incorporated into such popular protocols as PKIX-CMP and CMC.

4.1. PKIX-CMP

In PKIX-CMP, the ASN.1 [ASN1] construct, and corresponding comment for a certificate is given as follows.

```
CMPCertificate ::= CHOICE {
   x509v3PKCert        Certificate
}

-- This syntax, while bits-on-the-wire compatible with the
-- standard X.509 definition of "Certificate", allows the
-- possibility of future certificate types (such as X.509
-- attribute certificates, WAP WTLS certificates, or
-- other kinds of certificates) within this certificate
```

```
      -- management protocol, should a need ever arise to support
      -- such generality.
```

Building on this framework, this document expands the above CHOICE construct as follows.

```
      CMPCertificate ::= CHOICE {
         x509v3PKCert        Certificate,
         x509v2AttCert   [0] AttributeCertificate,
                             -- defined in [ATTCERT]
         openPGPCert     [2] OpenPGPCert
      }

      OpenPGPCert ::= OCTET STRING
         -- contains the OpenPGP certificate (OpenPGP Transferable
         -- Public Key) data structure from the OpenPGP specification
         -- [OPENPGP] (binary format without Radix-64 conversions),
         -- encoded as an ASN.1 OCTET STRING
```

Expanding the CHOICE construct as above allows X.509 attribute certificates and OpenPGP certificates to be used within the PKIX-CMP management messages. In the future, this construct may be expanded further (in subsequent revisions of this document) to accommodate other certificate types, if this is found to be necessary.

4.2. CMC

The CMC protocol uses the CMS (Cryptographic Message Syntax) syntax [CMS], which defines the certificate type as

```
     CertificateChoices ::= CHOICE {
       certificate Certificate,
       extendedCertificate [0] IMPLICIT ExtendedCertificate,  -- Obsolete
       v1AttrCert [1] IMPLICIT AttributeCertificateV1,        -- Obsolete
       v2AttrCert [2] IMPLICIT AttributeCertificateV2 }
```

Similar to PKIX-CMP, this CHOICE can be extended to include additional types of certificates as follows.

```
     CertificateChoices ::= CHOICE {
       certificate Certificate,
       extendedCertificate [0] IMPLICIT ExtendedCertificate,  -- Obsolete
       v1AttrCert [1] IMPLICIT AttributeCertificateV1,        -- Obsolete
       v2AttrCert [2] IMPLICIT AttributeCertificateV2,
       openPGPCert [3] IMPLICIT OpenPGPCert }
```

This allows both X.509 attribute certificates and OpenPGP certificates to be used within the CMC management messages. In the future, this construct may be expanded further (in subsequent revisions of this document) to accommodate other certificate types, if this is found to be necessary.

The CMC specification defines certain constraints on the subject and publicKey fields of the CRMF’s CertTemplate structure. The same constraints should apply to the AltCertTemplate structure if alternative certificate types are used. For example, the CMC specification mandates that

> When CRMF message bodies are used in the Full Enrollment Request message, each CRMF message MUST include both the subject and publicKey fields in the CertTemplate.

If alternative certificate types are used, this should be extended as

> When CRMF message bodies are used in the Full Enrollment Request message, each CRMF message MUST include both the subject and publicKey fields in the CertTemplate (or in the altCertTemplate control).

5. Security Considerations

5.1. Protection of Alternative Certificate Templates

This document defines extensions to the CRMF format, so security considerations from the CRMF specification [CRMF] apply here as well. In particular, the security of alternative certificate templates relies upon the security mechanisms of the protocol or process used to communicate with CAs.

Exact security requirements depend on a particular PKI deployment, but integrity protection and message origin authentication are typically required for certification requests. The CMP and CMC certificate management protocols mentioned in this document provide both integrity protection and message origin authentication for request messages (which includes certificate templates as well).

Confidentiality may also be required where alternative certificate templates contain subscriber-sensitive information. The CMC protocol allows the content of request messages to be encrypted. CMP does not include confidentiality mechanisms for certification requests, but if confidentiality is needed, it can be achieved with a lower-layer security protocol (e.g., TLS or IPsec).

5.2. Request Authorisation

In order to make a decision as to whether a request should be accepted, a CA should normally be able to compare the (authenticated) name of the sender of the request with the request subject name.

For example, an End Entity may be allowed to request additional certificates for himself/herself. In this case, the CA will accept a request if the Sender is equal to the Subject (of course, other conditions will have to be checked as well before the certificate is granted).

If a PGP certificate is requested using the extensions proposed here, the Sender field of the request will be encoded as an ASN.1 GeneralName (in both CMP and CMC), while the Subject will be represented as a PGP UserID. Since the PGP UserID is effectively an unrestricted octet string, it is not always trivial to compare these two types. It is possible that an attacker may try to submit requests with specially crafted UserIDs (e.g., that include obscure characters) in order to trick the CA comparison algorithm and obtain a PGP certificate with a UserID that belongs to someone else.

In these circumstances, it is safer for the CA, when building the PGP certificate’s UserID, to completely rebuild the UserID based on the content of the authenticated Sender name rather than take the UserID from the request. To achieve this, additional information about the End Entity may be required at the CA (e.g., the EE’s email address).

5.3. PGP Parser

Software components that implement the proposed extensions (e.g., CMP or CMC servers) will necessarily increase in complexity. If a "standard" server is expected to be able to parse ASN.1 streams, the "extended" server is required to be able to parse PGP streams as well. A PGP parser code may introduce new security vulnerabilities that can be exploited by an attacker to mount a DoS attack or gain access to the server.

In order to reduce the consequences of a successful attack, it is recommended that the CMP or CMC servers be run on a separate machine from the main CA server. These protocol servers should not have access to the main CA key and should not have write access to the CA store.

Appendix A. Examples of OpenPGP CertTemplates

This Appendix presents examples of OpenPGP CertTemplates that are used for requesting OpenPGP certificates from a CA.

A1. Simple Certificate Request

Alice requests an OpenPGP certificate for her public key accompanied by a subkey.

The content of the OpenPGP CertTemplate in the request is as follows. This CertTemplate conforms to the OpenPGP CertTemplate Required Profile.

```
0000:  99 01 A2          === Pub Key packet ===
0003:  04 3C 58 27 A2 11      ver 4, created 30 Jan 2002, DSA
0009:  00 E3 FB 9D .. 2B EF   DSA prime p
008B:  00 A0 FF 7E .. BA 71   DSA group order q
00A1:  03 FF 68 BC .. 56 71   DSA group generator g
0123:  03 FE 38 1F .. F2 63   DSA public key value y
01A5:  B4 19             === User ID packet ===
01A7:  41 6C .. 6D 3E         "Alice <alice@example.com>"
01C0:  89 00 49          === Signature packet (self-signature) ===
01C3:  04 10 11 02            ver 4, gen cert, DSA, SHA1
01C7:  00 09 05 02 3C 58 27 A2 02 1B 03
                              created 30 Jan 2002, key usage:
                              sign data and certify other keys
01D2:  00 0A 09 10 43 5C .. 06 77   issuer key id
01DE:  5A C2                  left 16 bits of signed hash value
01E0:  00 A0 EB 00 .. 1B 75   DSA value r
01F6:  00 A0 F4 E4 .. A8 3D   DSA value s
020C:  B9 02 0D          === Public Subkey packet ===
020F:  04 3C 58 27 A2 10      ver 4, created 30 Jan 2002,
                              Elgamal (encrypt-only) algorithm
0215:  08 00 F6 42 .. 0B 3B   Elgamal prime p
0317:  00 02 02               Elgamal group generator g
031A:  07 FE 37 BA .. DF 21   Elgamal public key value y
041C:  89 00 49          === Signature packet (subkey binding) ===
041F:  04 18 11 02            ver 4, subkey binding, DSA, SHA1
0423:  00 09 05 02 3C 58 27 A2 02 1B 0C
                              created 30 Jan 2002, key usage:
                              encrypt communications and storage
042E:  00 0A 09 10 43 5C .. 06 77   issuer key id
043A:  C7 DE                  left 16 bits of signed hash value
043C:  00 9E 21 33 .. 39 1B   DSA value r
0452:  00 9F 64 D7 .. 63 08   DSA value s
0468:
```

CA certifies Alice’s User ID and the public key and creates the following OpenPGP certificate:

```
0000:  99 01 A2             === Pub Key packet ===
0003:    <the same as in the request>
01A5:  B4 19            === User ID packet ===
01A7:    <the same as in the request>
01C0:  89 00 49         === Signature packet (self-signature) ===
01C3:    <the same as in the request>
020C:  89 00 49         === Signature packet (certification) ===
020F:  04 13 11 02            ver 4, positive cert, DSA, SHA1
0213:  00 09 05 02 3C 58 28 1A 02 1B 03
                              created 30 Jan 2002, key usage:
                              sign data and certify other keys
021E:  00 0A 09 10 F0 0D .. 1F CA   issuer key id
022A:  06 DF                  left 16 bits of signed hash value
022C:  00 9F 57 2D .. 26 E3   DSA value r
0242:  00 A0 B3 02 .. CE 65   DSA value s
0258:  B9 02 0D         === Public Subkey packet ===
025B:    <the same as in the request>
0468:  89 00 49         === Signature packet (subkey binding) ===
046B:    <the same as in the request>
04B4:
```

A2. Certificate Request with Central Key Generation

Alice requests that the CA generate an RSA key pair that will be used for signing, an RSA key pair that will be used for encryption, and requests that the CA certify these keys. The RSA keys are requested to be 2048 bits long with the public exponent 65537.

The content of the OpenPGP CertTemplate in the request is as follows:

```
0000:  99 01 0D         === Pub Key packet (Template) ===
0003:  04 FF FF FF FF 01      ver 4, any creation date, RSA
0009:  08 00 FF FF .. FF FF   RSA public modulus n - given length
010B:  00 11 01 00 01         RSA public exponent e
0110:  B4 19            === User ID packet ===
0112:  41 6C .. 6D 3E         "Alice <alice@example.com>"
012B:  89 00 23         === Signature packet (Template) ===
012E:  04 10 11 02            ver 4, gen cert, DSA, SHA1
0132:  00 09 05 02 FF FF FF FF 02 1B 03
                              any creation date, key usage:
                              sign data and certify other keys
013D:  00 0A 09 10 FF FF .. FF FF   issuer key id - any
0149:  05 3A                  left 16 bits of signed hash value
014B:  00 08 FF               DSA value r - any
014E:  00 08 FF               DSA value s - any
```

```
      0151:  99 01 0D          === Public Subkey packet (Template) ===
      0154:  04 FF FF FF FF 01      ver 4, any creation date, RSA
      015A:  08 00 FF FF .. FF FF   RSA public modulus n - given length
      025C:  00 11 01 00 01         RSA public exponent e
      0261:  89 00 20          === Signature packet (Template) ===
      0264:  04 18 01 02            ver 4, subkey binding, RSA, SHA1
      0268:  00 09 05 02 FF FF FF FF 02 1B 0C
                                    any creation date, key usage:
                                    encrypt communications and storage

      0273:  00 0A 09 10 FF FF .. FF FF   issuer key id - any
      027F:  12 E6                  left 16 bits of signed hash value
      0281:  00 08 FF               RSA signature value - any
      0284:
```

CA generates keys, certifies Alice’s User ID and the public key, and creates the following OpenPGP certificate:

```
      0000:  99 01 0D          === Pub Key packet  ===
      0003:  04 3C 5A A5 BB 01      ver 4, created 01 Feb 2002, RSA
      0009:  08 00 C7 21 .. 5B EB   RSA public modulus n
      010B:  00 11 01 00 01         RSA public exponent e
      0110:  B4 19             === User ID packet ===
      0112:  41 6C .. 6D 3E         "Alice <alice@example.com>"
      012B:  89 01 1F          === Signature packet (self-signature) ===
      012E:  04 10 01 02            ver 4, gen cert, RSA, SHA1
      0132:  00 09 05 02 3C 5A A5 BB 02 1B 03
                                    created 01 Feb 2002, key usage:
                                    sign data and certify other keys
      014D:  00 0A 09 10 8E AF .. 1A 18   issuer key id
      0149:  3B 21                  left 16 bits of signed hash value
      014B:  07 FE 2F 1D .. C0 81   RSA signature value
      024D:  89 00 49          === Signature packet (certification) ===
      0250:  04 13 11 02            ver 4, positive cert, DSA, SHA1
      0254:  00 09 05 02 3C 5A A5 DC 02 1B 03
                                    created 01 Feb 2002, key usage:
                                    sign data and certify other keys
      025F:  00 0A 09 10 F0 0D .. 1F CA   issuer key id
      026B:  BA C2                  left 16 bits of signed hash value
      026D:  00 9F 5E 58 .. 30 B3   DSA value r
      0283:  00 A0 D1 D7 .. 5A AF   DSA value s
      0299:  99 01 0D          === Public Subkey packet ===
      029C:  04 3C 5A A5 C5 01      ver 4, created 01 Feb 2002, RSA
      02A2:  08 00 C3 03 .. 8C 53   RSA public modulus n
      03A4:  00 11 01 00 01         RSA public exponent e
      03A9:  89 01 1F          === Signature packet (subkey binding) ===
      03AC:  04 18 01 02            ver 4, subkey binding, RSA, SHA1
      03B0:  00 09 05 02 3C 5A A5 C5 05 1B 0C
                                 created 01 Feb 2002, key usage:
                                 encrypt communications and storage
      03BB:  00 0A 09 10 8E AF .. 1A 18   issuer key id
      03C7:  C8 44                   left 16 bits of signed hash value
      03C9:  07 FB 04 D7 .. 75 BE   RSA signature value
      04CB:
```

Normative References

[ASN1] CCITT Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1), 1988.

[ATTCERT] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002.

[CMC] Myers, M., Liu, X., Schaad, J., and J. Weinstein, "Certificate Management Messages over CMS", RFC 2797, April 2000.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

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

[CRMF] Schaad, J., "Internet X.509 Public Key Infrastructure: Certificate Request Message Format (CRMF)", RFC 4211, September 2005.

[OPENPGP] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

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

Authors' Addresses

Mikhail Blinov
Guardeonic Solutions
Fitzwilliam Court, Leeson Close
Dublin 2, Ireland

EMail: mikblinov@online.ie

Carlisle Adams
School of Information Technology and Engineering (SITE)
University of Ottawa
800 King Edward Avenue
P.O. Box 450, Stn A
Ottawa, Ontario, Canada K1N 6N5

EMail: cadams@site.uottawa.ca

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.