Workgroup: LAMPS
Internet-Draft:
draft-truskovsky-lamps-pq-hybrid-x509-02
Published: 24 August 2023
Intended Status: Standards Track
Expires: 25 February 2024
Authors: A. Truskovsky D. Van Geest
ISARA Corporation ISARA Corporation
S. Fluhrer P. Kampanakis
Cisco Systems Cisco Systems
M. Ounsworth S. Mister
Entrust Datacard, Ltd Entrust Datacard, Ltd

Multiple Public-Key Algorithm X.509 Certificates

Abstract

Tombstone notice:

This draft is no longer being pursued at the IETF. A variant of this proposal was adopted in [itu-t-x509-2019], which allows two keys to be placed in a certificate but only one used at a time. The major downside of this proposal is that it requires the large PQC key to be sent even to legacy clients which will not use it. Additionally, this proposal does not present a generic encoding for the multiple signatures produced by the multiple keys contained in a hybrid certificate, leaving the responsibility to dependent protocols and applications for how to carry multiple signatures and how to signal that multiple signatures should have been present in order to detect stripping attacks. As such, this document represents only a partial solution to the dual-signature problem. How, and whether, to implement dual-signatures is an active and ongoing discussion topic at the IETF and work continues in this area across several working groups. The PQUIP WG serves as a central location for all PQC-related discussion.

Original abstract:

This document describes a method of embedding alternative sets of cryptographic materials into X.509v3 digital certificates, X.509v2 Certificate Revocation Lists (CRLs), and PKCS #10 Certificate Signing Requests (CSRs). The embedded alternative cryptographic materials allow a Public Key Infrastructure (PKI) to use multiple cryptographic algorithms in a single object, and allow it to transition to the new cryptographic algorithms while maintaining backwards compatibility with systems using the existing algorithms. Three X.509 extensions and three PKCS #10 attributes are defined, and the signing and verification procedures for the alternative cryptographic material contained in the extensions and attributes are detailed.

Status of This Memo

This Internet-Draft will expire on 25 February 2024.

Copyright Notice

Table of Contents

1. Introduction

Modern Public Key Infrastructure (PKI) extensively relies on classical signature algorithms such as RSA or ECDSA to achieve secure authentication. The security of these algorithms is based on the time-tested difficulty of certain number-theoretic problems. However, it is well known that such schemes offer insufficient security against an adversary in possession of a universal quantum computer. Such an adversary can efficiently recover the private key from the public key and impersonate any entity in the system -- even a root Certification Authority (CA). Hence, it is necessary to upgrade these PKIs to utilize algorithms that are secure against such adversaries.

An obvious solution is for relying parties to require multiple certificates to establish trust in an entity. One could theoretically continue to use certificates as they currently are and introduce separate certificates that utilize the new algorithms. However, managing different cryptographic algorithms within a single PKI in this way requires multiple certificate chains. This would greatly increase the complexity of the already complex system. Furthermore, some systems rely on physical solutions for credential storage. These physical solutions may be limited in terms of capacity as well as in terms of how such systems are interacted with. Instead, it is far simpler to keep only a single identity and employ a single certificate chain for each user.

The goal of this document is to profile new X.509v3 certificate extensions, X.509v2 CRL extensions and PKCS #10 CSR attributes that facilitate the use of a simple and efficient approach for executing this upgrade. A key design requirement for this approach is to not affect the behavior of non-upgraded systems and ensure they can process any new attributes or extensions without breaking.

By placing an alternative public key and alternative signature into custom extensions, one effectively embeds multiple certificate chains within a single chain. By utilizing these multiple public-key algorithm certificates, legacy applications can continue using their current choices of cryptographic algorithms and upgraded applications can use new algorithms while remaining interoperable with the legacy systems.

It is useful to observe that even though the motivation for this document is to upgrade PKIs to use quantum-safe cryptography, the same methodology can be used to upgrade such systems to any new algorithm. For this reason, this document does not specify that quantum-safe algorithms are the new technology the PKI is being upgraded to use.

The remainder of this document is organized as follows.

Section 2 profiles the three new PKCS #10 attributes and three new X.509 extensions. Sections 3, 4 and 5 profile methods for signing and verifying CSRs, certificates and CRLs respectively using the new extensions.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Terminology

The following terms are defined:

- alternative algorithm: The algorithm, whose usage is profiled in this document, which can be used to sign and verify a certificate instead of, or in addition to, the conventional algorithm.

- alternative [public, private] key: The keys, whose usage is profiled in this document, which can be used to create or verify a signature instead of, or in addition to, the conventional keys.

- alternative signature: The signature, whose usage is profiled in this document, which can be used to validate a certificate instead of, or in addition to, the conventional signature.

*conventional algorithm: The algorithm specified in the signatureAlgorithm field of an X.509v3 certificate.

*conventional [public, private] key: The key used to create or verify a conventional signature in an X.509v3 certificate.

*conventional signature: The value specified in the signature field of an X.509v3 certificate.

*multiple public-key algorithm certificate: A certificate which is equipped with the extensions introduced in this document. Thus, the certificate is signed and can be verified using two different public-key algorithms. One public-key algorithm (the "conventional" one) uses the keys, signatures and algorithms specified in the standard X.509v3 fields. The other ("alternative") public-key algorithm uses the keys, signatures and algorithms in the extensions defined in this document.

*upgraded [application, system]: An application or system which is capable of understanding and using the extensions introduced in this document.

2. Alternative Public-Key Algorithm Objects

2.1. OIDs

The following OIDs are used to identify the CSR attributes and X. 509v3 extensions defined in the following sections.

```
id-subjectAltPublicKeyInfo  OBJECT IDENTIFIER  ::=  { TBD }

id-altSignatureAlgorithm    OBJECT IDENTIFIER  ::=  { TBD }

id-altSignatureValue        OBJECT IDENTIFIER  ::=  { TBD }
```

2.2. CSR Attributes

Three new CSR attributes are used to submit an alternative public key for certification. Each of these attributes mirror existing fields within a CSR and serve the same purpose as those fields, but with the alternative algorithms. An entity creating a CSR MUST include either all three of these attributes or none.

2.2.1. Subject Alt Public Key Info Attribute

The Subject Alt Public Key Info Attribute corresponds to the SubjectPublicKeyInfo type defined in Section 4.1 of [RFC2986]. This attribute carries information about the alternative public key being certified. The information also identifies the entity's alternative public-key algorithm (and any associated parameters).

This attribute is identified using the id-subjectAltPublicKeyInfo OID.

```
SubjectAltPublicKeyInfoAttr ATTRIBUTE  ::=  {
     WITH SYNTAX SubjectPublicKeyInfo
     ID id-subjectAltPublicKeyInfo }
```

2.2.2. Alt Signature Algorithm Attribute

The Alt Signature Algorithm attribute corresponds to the signatureAlgorithm field of the CertificationRequest type described in Section 4.2 of [RFC2986]. This attribute contains the identifier for the alternative cryptographic algorithm used by the requesting entity to sign the CertificationRequestInfo.

This attribute is identified using the id-altSignatureAlgorithm OID.

```
AltSignatureAlgorithmAttr ATTRIBUTE  ::=  {
     WITH SYNTAX AlgorithmIdentifier
     ID id-altSignatureAlgorithm }
```

2.2.3. Alt Signature Value Attribute

The Alt Signature Value attribute corresponds to the signature field of the CertificationRequest type described in Section 4.2 of [RFC2986]. This attribute contains a digital signature computed upon the ASN.1 DER encoded PreCertificationRequestInfo as described in Section 3 of this document.

By generating this alternative signature, a certification request subject proves possession of the alternative private key.

This attribute is identified using the id-altSignatureValue OID.

```
AltSignatureValueAttr ATTRIBUTE  ::=  {
     WITH SYNTAX BIT STRING
     EQUALITY MATCHING RULE bitStringMatch
     ID id-altSignatureValue }
```

2.3. X.509v3 Extensions

Three new X.509v3 extensions are used to authenticate a certificate using alternative algorithms. Each of these extensions mirror existing fields within an X.509v3 certificate and serve the same purpose as those fields, but with the alternative algorithms.

2.3.1. Subject Alt Public Key Info Extension

The Subject Alt Public Key Info extension corresponds to the Subject Public Key Info field described in Section 4.1.2.7 of [RFC5280].

This extension carries the alternative public key, and identifies the algorithm with which the key is used.

This extension is identified using the id-subjectAltPublicKeyInfo OID.

```
SubjectAltPublicKeyInfoExt  ::=  SEQUENCE  {
     algorithm                   AlgorithmIdentifier,
     subjectAltPublicKey         BIT STRING }
```

2.3.2. Alt Signature Algorithm Extension

The Alt Signature Algorithm extension corresponds to the signature field described in Section 4.1.2.3 of [RFC5280]. It also corresponds to the signatureAlgorithm field described in Section 4.1.1.2 of [RFC5280] since both those fields have the same values. This extension contains the identifier for the alternative digital signature algorithm used by the CA to sign the preTBSCertificate.

This extension is identified using the id-altSignatureAlgorithm OID.

```
AltSignatureAlgorithmExt  ::=  AlgorithmIdentifier
```

2.3.3. Alt Signature Value Extension

The Alt Signature Value extension corresponds to the signatureValue field described in Section 4.1.1.3 of [RFC5280]. This extension contains a digital signature computed upon the ASN.1 DER encoded preTBSCertificate as described in Section 4.

By generating this alternative signature, a CA certifies the validity of the preTBSCertificate data. In particular, the CA certifies the binding between the alternative public key material and the subject of the certificate.

This extension is identified using the id-altSignatureValue OID.

```
AltSignatureValueExt  ::=  BIT STRING
```

3. Multiple Public-Key Algorithm Certificate Signing Requests

A Certificate Signing Request (CSR) is a sequence of three required fields as defined in Section 4.2 of [RFC2986].

```
CertificationRequest  ::=  SEQUENCE  {
     certificationRequestInfo    CertificationRequestInfo,
     signatureAlgorithm          AlgorithmIdentifier,
     signature                   BIT STRING }
```

A CSR's signature is calculated on the ASN.1 DER encoding of the CertificationRequestInfo object as defined in Section 4.2 of [RFC2986].

```
CertificationRequestInfo  ::=  SEQUENCE  {
     version       INTEGER  { v1(0) } (v1,...),
     subject       Name,
     subjectPKInfo SubjectPublicKeyInfo{{ PKInfoAlgorithms }},
     attributes    [0] Attributes{{ CRIAttributes }} }
```

The alternative signature is calculated on the ASN.1 DER encoding of the identical PreCertificationRequestInfo object.

```
PreCertificationRequestInfo  ::=  SEQUENCE  {
     version       INTEGER  { v1(0) } (v1,...),
     subject       Name,
     subjectPKInfo SubjectPublicKeyInfo{{ PKInfoAlgorithms }},
     attributes    [0] Attributes{{ CRIAttributes }} }
```

The PreCertificationRequestInfo type is the same as the CertificationRequestInfo type, however the PreCertificationRequestInfo object will have different attributes than the CertificationRequestInfo. Specifically, the CertificationRequestInfo will include the AltSignatureValueAttr attribute, while the PreCertificationRequestInfo will not.

3.1. Creating Multiple Public-Key Algorithm CSRs

A multiple public-key algorithm CSR requires the applicant to generate two key pairs: one for the old algorithm (the conventional key pair), and another for the new algorithm (the alternative key pair). All actions taken by the applicant with regards to the conventional algorithm and key pair are unchanged during this process. Additional attributes are populated to prove that the applicant is in possession of the alternative private key.

The PreCertificationRequestInfo object MUST contain the SubjectAltPublicKeyInfoAttr attribute carrying the alternative public key and algorithm for the CSR being created.

The PreCertificationRequestInfo object MUST contain the AltSignatureAlgorithmAttr attribute, which specifies the algorithm identifier for the algorithm used to sign the PreCertificationRequestInfo object.

The alternative signature of the PreCertificationRequestInfo MUST be calculated using the alternative private key of the certificate request subject, which is the private key associated with the public key found in the subject's SubjectAltPublicKeyInfoAttr attribute.

After the alternative signature is calculated, the alternative signature MUST be added as an AltSignatureValueAttr attribute to create the CertificationRequestInfo object.

The process of signing a multiple public-key algorithm CSR as described above can be summarized as follows:

a. Create a PreCertificationRequestInfo object, which is populated with all the data to be signed by the alternative private key, including the SubjectAltPublicKeyInfoAttr and AltSignatureAlgorithmAttr attributes.

b. Calculate the alternative signature on the DER encoding of the PreCertificationRequestInfo, using the certificate request subject's alternative private key with the algorithm specified in the AltSignatureAlgorithmAttr attribute.

c. Convert the PreCertificationRequestInfo to a CertificationRequestInfo by adding the calculated alternative signature to the PreCertificationRequestInfo object using the AltSignatureValueAttr attribute.

d. As per [RFC2986], calculate the conventional signature using the certificate request subject's conventional private key and create the CertificationRequest from the certificationRequestInfo, signatureAlgorithm and signature.

An upgraded system MAY issue both multiple public-key algorithm and single public-key algorithm CSRs depending on their policies. If the system issues a single public-key algorithm CSR, then that CSR MUST NOT contain any of the three attributes profiled in this section.

3.2. Verifying Multiple Public-Key Algorithm CSRs

The certificate issuer verifies the alternative signature of the multiple public-key algorithm CSR by reconstructing the PreCertificationRequestInfo object and using its ASN.1 DER encoding, alternative public key and alternative signature algorithm to verify the signature.

To verify the alternative signature of a multiple public-key algorithm CSR, the following steps are taken:

a. ASN.1 DER decode the certificationRequestInfo field of the CertificationRequest to get a CertificationRequestInfo object.

b. Remove the AltSignatureValueAttr attribute from the CertificationRequestInfo object and set aside the alternative signature. The object is now the same as the

PreCertificationRequestInfo which the signature was generated on.

c. ASN.1 DER encode the PreCertificationRequestInfo object.

d. Using the algorithm specified in the AltSignatureAlgorithmAttr attribute of the PreCertificationRequestInfo, the alternative public key from the CSR's SubjectAltPublicKeyInfoAttr attribute and the ASN.1 DER encoded PreCertificationRequestInfo, verify the alternative signature from (b).

During the process of ASN.1 DER decoding the CertificationRequestInfo, removing the AltSignatureValueAttr attribute from the PreCertificationRequestInfo, and ASN.1 DER encoding the PreCertificationRequestInfo, the relative ordering of the remaining attributes is not modified. This is due to the DER encoding rules applied during signature generation as specified in RFC2986. Thus, the resulting ASN.1 DER encoded PreCertificationRequestInfo is identical to the one the issuer used to generate the alternative signature.

4. Multiple Public-Key Algorithm Certificates

An X.509 digital certificate is a sequence of three fields as defined in [RFC5280].

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING }
```

An X.509v3 certificate's signature is calculated on the ASN.1 DER encoding of the TBSCertificate object as defined in Section 4.1 of [RFC5280]. In this way, a CA certifies the validity of the information in the tbsCertificate field, in particular the binding between the conventional public key material and the subject of the certificate.

The alternative signature is calculated on the ASN.1 DER encoding of the similar, but not identical, PreTBSCertificate defined below. This signature also certifies the validity of the information in the tbsCertificate field. In particular, the binding between the alternative public key material and the subject of the certificate is validated.

```
PreTBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }
```

The PreTBSCertificate type is similar to the TBSCertificate type, except that the PreTBSCertificate does not include the signature field (the third element in the TBSCertificate sequence). In a TBSCertificate the signature field contains the AlgorithmIdentifier of the algorithm which will be used to sign the final certificate, and this value might not be known at the time that the alternative signature is calculated. Additionally, since the AlgorithmIdentifier of the signature field is associated with the final signatureValue field in the certificate, it is outside the scope of the alternative public-key algorithm and does not need to be protected by the alternative signature.

The PreTBSCertificate object also does not contain the AltSignatureValueExt extension in its extension list, while the TBSCertificate will. Since the alternative signature is calculated on the encoding of the PreTBSCertificate it cannot be included in the PreTBSCertificate.

4.1. Creating Multiple Public-Key Algorithm Certificates

If a CA is issuing a subject certificate and the issuer certificate or root of trust contains an alternative public key, then the CA SHOULD add an alternative signature to the subject certificate. Failure to do so could result in a verifier rejecting the certificate as being malformed, especially if the verifier is concerned about quantum-enabled adversaries. This is discussed further in Section 8.1.

A multiple public-key algorithm certificate MAY contain the SubjectAltPublicKeyInfoExt extension. If the certificate's subject has an alternative public key which they wish to bind to their identity, then the public key and algorithm MUST be placed in the SubjectAltPublicKeyInfoExt extension. However, if the certificate's subject has no such alternative public key (e.g. the subject's application has not been upgraded to support multiple public-key algorithms) then the SubjectAltPublicKeyInfoExt extension will not be added to the certificate.

If a CA is issuing a certificate with an alternative signature, the extensions field of the PreTBSCertificate MUST contain the AltSignatureAlgorithmExt extension, which specifies the algorithm identifier for the algorithm used to sign the PreTBSCertificate.

The alternative signature of the PreTBSCertificate MUST be calculated using the alternative private key of the Issuer, which is the private key associated with the public key found in the Issuer's SubjectAltPublicKeyInfoExt extension.

After the alternative signature is calculated, the alternative signature MUST be added as an AltSignatureValueExt extension to the extensions list of the PreTBSCertificate, resulting in the TBSCertificate.

The process of signing an X.509v3 multiple public-key algorithm certificate as described above can be summarized as follows:

a. Create a PreTBSCertificate object, which is populated with all the data to be signed by the alternative private key, including the SubjectAltPublicKeyInfoExt and AltSignatureAlgorithmExt extensions.

b. Calculate the alternative signature on the DER encoding of the PreTBSCertificate, using the Issuer's alternative private key with the algorithm specified in the AltSignatureAlgorithmExt extension.

c. Add the calculated alternative signature to the PreTBSCertificate object using the AltSignatureValueExt extension.

d. Convert the PreTBSCertificate to a TBSCertificate by adding the signature field and populating it with the algorithm identifier of the conventional algorithm to be used to sign the certificate.

e. As per [RFC5280], calculate the conventional signature using the conventional private key associated with the Issuer's certificate and create the certificate from the tbsCertificate, signatureAlgorithm and signature.

If the upgraded CA's policy allows it to process single public-key algorithm CSRs and issue single public-key algorithm certificates, and the issuer's certificate has an alternative public key, and the CA receives a single-algorithm CSR, the CA SHOULD still include properly calculated AltSignatureValueExt and AltSignatureAlgorithmExt extensions in the certificate. This ensures that when an upgraded system verifies the subject's certificate and sees that the issuer certificate contains the SubjectAltPublicKeyInfoExt extension that it will verify the subject's alternative signature. Otherwise it might treat the subject's certificate as invalid. This is discussed further in the Security Considerations section.

Note - A certificate issuer would typically mark the SubjectAltPublicKeyInfoExt, AltSignatureAlgorithmExt and AltSignatureValueExt extensions as non-critical, allowing the multiple public-key algorithm certificate to be treated like a regular certificate by non-upgraded entities. However, the issuer MAY mark the extensions as critical, for example if it is part of a PKI which requires entities to understand both the conventional and alternative signatures.

4.2. Verifying Multiple Public-Key Algorithm Certificates

Users wishing to verify a multiple public-key algorithm certificate using the alternative public-key algorithm will need to convert the tbsCertificate field in the certificate to a PreTBSCertificate object identical to the PreTBSCertificate object which the issuer used to create the alternative signature. Then the user can use the issuer's alternative public key with the alternative signature algorithm to verify the alternative signature of the PreTBSCertificate.

To verify the alternative signature of the multiple public-key algorithm certificate, the following steps are taken:

a. ASN.1 DER decode the tbsCertificate field of the certificate to get a TBSCertificate object.

b. Remove the AltSignatureValueExt extension from the TBSCertificate object and set aside the alternative signature.

c. Remove the signature field from the TBSCertificate object, converting it to a PreTBSCertificate object.

d. ASN.1 DER encode the PreTBSCertificate object.

e. Using the algorithm specified in the AltSignatureAlgorithmExt extension of the PreTBSCertificate, the alternative public key from the Issuer's SubjectAltPublicKeyInfoExt extension and the ASN.1 DER encoded PreTBSCertificate, verify the alternative signature from (b).

The issuer's alternative public key comes from the issuing certificate's SubjectAltPublicKeyInfoExt extension, unless the issuer is a trust anchor. In that case, the trust anchor's alternative public key may come from a self-signed certificate's SubjectAltPublicKeyInfoExt extension, or it may come from elsewhere. [RFC5280] section 6.1.1 (d) lists the trust anchor information as including:

a. the trusted issuer name,

b. the trusted public key algorithm,

c. the trusted public key, and

d. optionally, the trusted public key parameters associated with the public key.

When validating a multiple public-key algorithm certificate, the trust anchor information also includes:

a. the trusted alternative public key algorithm,

b. the trusted alternative public key, and

c. optionally, the trusted alternative public key parameters associated with the alternative public key.

During the process of ASN.1 DER decoding the TBSCertificate, removing the AltSignatureValueExt extension from the PreTBSCertificate and ASN.1 DER encoding the PreTBSCertificate, the relative ordering of the remaining extensions is not modified. Thus, the resulting ASN.1 DER encoded PreTBSCertificate is identical to the one the issuer used to generate the alternative signature.

A certificate that contains an AltSignatureValueExt extension but does not contain an AltSignatureAlgorithmExt extension cannot be verified under the alternative public-key algorithm and so SHOULD be rejected as being malformed. Similarly, a certificate that contains an AltSignatureAlgorithmExt extension but does not contain an AltSignatureValueExt extension SHOULD be rejected.

A certificate MAY have AltSignatureValueExt and AltSignatureAlgorithmExt extensions without having a SubjectAltPublicKeyInfoExt extension. This case could arise if a non-upgraded subject requests a certificate from an upgraded CA who has a multiple public-key algorithm CA certificate.

If an issuer certificate or root of trust has an alternative public key, but a subject certificate issued by the issuer certificate or root of trust doesn't contain an alternative signature then the verifier SHOULD reject the subject certificate. This is especially important if the verifier is concerned about quantum-enabled adversaries. This is discussed further in the Section 8.1. Accepting such a subject certificate SHOULD be limited to cases where the verifier has been explicitly configured to ignore missing alternative signatures for a given issuing CA, for subject certificates matching a given wildcard, or similar whitelisting mechanisms.

5. Multiple Public-Key Algorithm Certificate Revocation Lists

In certain situations, certificates must be revoked and no longer used. This can happen for a variety of reasons including, but not limited to: key compromise, CA compromise, or due to a change in affiliation. Roughly speaking, Certificate Revocation Lists (CRLs) are authenticated lists of revoked certificates.

An X.509v2 Certificate Revocation List (CRL) is a sequence of three fields as defined in [RFC5280].

```
CertificateList  ::=  SEQUENCE  {
    tbsCertList          TBSCertList,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValue       BIT STRING  }
```

An X.509v2 CRL's signature is calculated on the ASN.1 DER encoding of the TBSCertList object as defined in Section 5.1 of [RFC5280].

The alternative signature is calculated on the ASN.1 DER encoding of the similar, but not identical, PreTBSCertList object defined here.

```
PreTBSCertList  ::=  SEQUENCE  {
     version                 Version OPTIONAL,
                                  -- if present, MUST be v2
     issuer                  Name,
     thisUpdate              Time,
     nextUpdate              Time OPTIONAL,
     revokedCertificates     SEQUENCE OF SEQUENCE  {
          userCertificate         CertificateSerialNumber,
          revocationDate          Time,
          crlEntryExtensions      Extensions OPTIONAL
                                   -- if present, version MUST be v2
                               }  OPTIONAL,
     crlExtensions           [0]  EXPLICIT Extensions OPTIONAL
                                   -- if present, version MUST be v2
                               }
```

The PreTBSCertList object is similar to the TBSCertList object, except that the PreTBSCertList does not include the signature field (the second element in the TBSCertList sequence). In a TBSCertList the signature field contains the AlgorithmIdentifier of the algorithm which will sign the final certificate revocation list, and this value might not be known at the time that the alternative signature is calculated. Additionally, since the AlgorithmIdentifier of the signature field is associated with the final signatureValue field in the CRL, it is outside the scope of the alternative public-key algorithm and does not need to be protected by the alternative signature.

The PreTBSCertList object also does not contain the AltSignatureValueExt extension in its extension list, while the TBSCertList will. Since the alternative signature is calculated on the encoding of the PreTBSCertList, it cannot be included in the TBSCertList.

If a multiple public-key algorithm certificate is revoked, whether because the classical key is compromised, the alternative key is compromised or or other reason, both the classical and alternative keys SHOULD be considered revoked. This avoids any unneeded complexity in dealing with a certificate where one key is compromised but the other isn't.

5.1. Creating Multiple Public-Key Algorithm Certificate Revocation Lists

To create a multiple public-key algorithm CRL, one creates a CRL as specified in Section 5 of [RFC5280] and includes the additional extensions as specified in this section.

If the CRL issuer's certificate has a SubjectAltPublicKeyInfoExt extension, the CRL SHOULD be created with an alternative signature. Otherwise, some upgraded systems may fail to validate the CRL because it is not trusted under the alternative public-key algorithm.

The extensions field of the PreTBSCertList MUST contain the AltSignatureAlgorithmExt extension, which specifies the algorithm identifier for the algorithm used to sign the PreTBSCertList.

The alternative signature of the PreTBSCertList MUST be calculated using the alternative private key of the CRL issuer, which is the private key associated with the public key found in the CRL issuer X.509v3 certificate's SubjectAltPublicKeyInfoExt extension.

After the alternative signature is calculated, the alternative signature MUST be added as an AltSignatureValueExt extension to the extensions list of the PreTBSCertList, resulting in the TBSCertList.

The process of signing an X.509v2 multiple public-key algorithm CRL as described above can be summarized as follows:

a. Create a TBSCertList object, which is populated with all the data to be signed by the alternative private key, including the AltSignatureAlgorithmExt extension.

b. Calculate the alternative signature on the DER encoding of the PreTBSCertList, using the CRL issuer's alternative private key with the algorithm specified in the AltSignatureAlgorithmExt extension.

c. Add the calculated alternative signature to the PreTBSCertList object using the AltSignatureValueExt extension.

d. Convert the PreTBSCertList to a TBSCertList by adding the signature field and populating it with the algorithm identifier of the conventional algorithm to be used to sign the certificate.

e. As per [RFC5280], calculate the conventional signature using the conventional private key associated with the CRL issuer's certificate and create the CRL from the tbsCertList, signatureAlgorithm and signature.

Note - A CRL issuer would typically mark the AltSignatureAlgorithmExt and AltSignatureValueExt extensions as non-critical, allowing the multiple public-key algorithm CRL to be treated like a regular CRL by non-upgraded entities. However, the issuer may be part of a PKI which requires entities to understand both the conventional and alternative signatures, in which case it would mark the extensions as critical.

5.2. Verifying Multiple Public-Key Algorithm Certificate Revocation Lists

Users wishing to verify the alternative signature of a multiple public-key algorithm CRL will need to convert the tbsCertList field in the CRL to a PreTBSCertList identical to the PreTBSCertList which the issuer used to create the alternative signature. Then the user can use the CRL issuer certificate's alternative public key with the alternative signature algorithm to verify the alternative signature of the PreTBSCertList.

To verify the alternative signature of the multiple public-key algorithm CRL, the following steps are taken:

a. ASN.1 DER decode the tbsCertList field of the certificate to get a TBSCertList object.

b. Remove the AltSignatureValueExt extension from the TBSCertList object and set aside the alternative signature.

c. Remove the signature field from the TBSCertList object, converting it to a PreTBSCertList object.

d. ASN.1 DER encode the PreTBSCertList object.

e. Using the algorithm specified in the AltSignatureAlgorithmExt extension of the PreTBSCertList, the alternative public key from the CRL issuer certificate's SubjectAltPublicKeyInfoExt extension and the ASN.1 DER encoded PreTBSCertList, verify the alternative signature from (b).

During the process of ASN.1 DER decoding the TBSCertList, removing the AltSignatureValueExt extension from the PreTBSCertList and ASN.1 DER encoding the PreTBSCertList, the relative ordering of the remaining extensions will not be modified. Thus, the resulting ASN.1 DER encoded PreTBSCertList is identical to the one the issuer used to generate the alternative signature.

In addition to verifying the alternative signature of a CRL, an implementation also needs to validate the CRL issuer's certificate and the certificate chain it is a part of. Implementations SHOULD use the same method as profiled in Section 6 of [RFC5280] with the following modifications to the CRL processing algorithm of that document's Section 6.3.3. Step (f) of the CRL processing algorithm requires certificate path validation for the issuer of the complete CRL. To validate multiple public-key algorithm CRLs, upgraded entities SHOULD additionally verify the alternative signatures along the path as described in Section 4.2 of this document. Step (g) of the CRL Processing algorithm requires the verification of a single signature on the complete CRL. To verify multiple public-key algorithm CRLs, this step MUST be modified to instead verify dual signatures on the complete CRL. Similarly, in step (h) of the same algorithm, if use-deltas is set and if the delta CRL is a multiple public-key algorithm CRL, then the verifying peer should validate the signature on the delta CRL via the method described above, and use standard practice otherwise - using the public key(s) validated in step (f).

6. Acknowledgements

The authors would like to thank Philip Lafrance and John Gray for their valuable contributions.

7. IANA Considerations

Extensions in certificates and CRLs are identified using object Identifiers (OIDs). The creation and delegation of these arcs is to be determined.

8. Security Considerations

Many of the security considerations for this document closely follow those of [RFC5280]. However, the use of the extensions introduced in this document does bring rise to additional considerations.

The motivation behind this document is to provide a method of upgrading PKIs and dependent systems to achieve quantum-safe state. However, state-of-the-art quantum-safe signature schemes tend to have large signature or key sizes. As such, their inclusion on CSRs, certificates, or CRLs means that the sizes of these data structures will significantly increase. This could potentially cause problems in protocols or implementations expecting more reasonable sizes. Even if enterprises choose instead to upgrade their PKI to new, but still classically secure signature algorithms, these algorithms can also be expected to have large signature or key sizes; often a by-product of an increased level of security is larger signatures or key sizes.

There is a great deal of flexibility inherent to the use of the extensions introduced in this document. Their design is such that a clean separation is made between the old and new signatures. The new signatures have no dependency on the old signatures and no understanding of the new signatures is required to compute or verify the old signature. As such, one could rely on the conventional signature only, the alternative signature only, or both, depending on the policies of the entity.

It is paramount that all private keying material be kept secret; a subject covered in the Security Considerations section of [RFC5280]. If the PKI is upgraded to use quantum-safe technologies, then it is of key importance to ensure that all private materials are protected against quantum-enabled adversaries as well. How such a feat is accomplished is outside the scope of this document. Additionally, issues such as re-keying or key management are outside the scope of this document.

Typically, the SubjectAltPublicKeyInfoExt, AltSignatureAlgorithmExt and AltSignatureValueExt extensions will be marked as non-critical so that a non-upgraded system could treat a multiple public-key algorithm certificate or CSR as a conventional certificate. However, a PKI could choose to enforce the usage of both conventional and alternative public-key algorithms, in which case it MAY mark these extensions as critical. The reasons why a PKI may want to do this are outside the scope of this document.

8.1. Post-Quantum Security Considerations

While this document is intended to facilitate transitioning a PKI from a classical public-key algorithm to a quantum-safe public-key algorithm, with the transition completing before the development of quantum computers capable of breaking classical public-key algorithms, it is worth discussing security considerations if multiple public-key algorithm certificates are used in the presence of a quantum-enabled adversary.

A quantum-enabled adversary is expected to be able to forge signatures for certificates and CRLs using classically secure signature algorithms. Thus, a CA SHOULD add an alternative signature to any certificate it issues if the issuing certificate contains a SubjectAltPublicKeyInfoExt extension. If the trust anchor is not a certificate, the alternative signature SHOULD be added if the trust anchor has an associated alternative public key which could be used for verification. Similarly, when verifying certificates or CRLs an application SHOULD reject certificates or CRLs if they don't contain an alternative signature but the issuer certificate does contain a SubjectAltPublicKeyInfoExt or the trust anchor has an alternative public key. If the CA does not add the alternative signature in these cases, and an upgraded application does not take this precaution when verifying, then a quantum-enabled adversary could create a certificate or CRL without an alternative signature, and forge the conventional signature of any issuer, causing upgraded applications to accept forged credentials.

If an upgraded relying party processing a non-multiple public-key algorithm CRL encounters a multiple public-key algorithm certificate (containing an AltSignatureValueExt extension) in the list of revoked certificates, it SHOULD NOT treat that certificate as revoked. If the conventional signature of the CRL uses a non-quantum-safe signature algorithm (e.g. RSA or ECDSA), a quantum-enabled attacker may have forged the CRL, thereby revoking certificates that the CA didn't intend to revoke. If one of those certificates has the multiple public-key algorithm extension then it was intended to be processed using the alternative public-key algorithm and should not be revoked based on only the results of the conventional public-key algorithm.

9. References

9.1. Normative References

[itu-t-x509-2019]

ITU-T, "ITU-T X.509 The Directory: Public-key and attribute certificate frameworks", January 2019, <https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.509>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

9.2. Informative References

[I-D.becker-guthrie-cert-binding-for-multi-auth] Becker, A., Guthrie, R., and M. J. Jenkins, "Related Certificates for Use in Multiple Authentications within a Protocol", Work in Progress, Internet-Draft, draft-becker-guthrie-cert-binding-for-multi-auth-02, 5 January 2023, <https://datatracker.ietf.org/doc/html/draft-becker-guthrie-cert-binding-for-multi-auth-02>.

[I-D.bonnell-lamps-chameleon-certs] Bonnell, C., Gray, J., Hook, D., Okubo, T., and M. Ounsworth, "A Mechanism for Encoding Differences in Paired Certificates", Work in Progress, Internet-Draft, draft-bonnell-lamps-chameleon-certs-01, 28 June 2023, <https://datatracker.ietf.org/doc/html/draft-bonnell-lamps-chameleon-certs-01>.

[I-D.ounsworth-pq-composite-sigs] Ounsworth, M., Gray, J., and M. Pala, "Composite Signatures For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-sigs-09, 29 May 2023, <https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-sigs-09>.

Appendix A. ASN.1 Structures and OIDs

This appendix includes all of the ASN.1 type and value definitions introduced in this document.

```
DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All --
-- IMPORTS NONE --

-- Object Identifiers for the certificate extensions introduced in
-- Section 4.

id-subjectAltPublicKeyInfo OBJECT IDENTIFIER  ::=  { TBD }

id-altSignatureAlgorithm OBJECT IDENTIFIER  ::=  { TBD }

id-altSignatureValue OBJECT IDENTIFIER  ::=  { TBD }

-- X.509 Certificate extensions

SubjectAltPublicKeyInfoExt  ::=  SEQUENCE  {
     algorithm                 AlgorithmIdentifier,
     subjectAltPublicKey        BIT STRING }

AltSignatureAlgorithmExt  ::=  AlgorithmIdentifier

AltSignatureValueExt  ::=  BIT STRING

-- attribute data types

subjectAltPublicKeyInfoAttr ATTRIBUTE  ::=  {
     WITH SYNTAX SubjectPublicKeyInfo
     ID id-subjectAltPublicKeyInfo }

altSignatureAlgorithmAttr ATTRIBUTE  ::=  {
     WITH SYNTAX AlgorithmIdentifier
     ID id-altSignatureAlgorithm }

altSignatureValueAttr ATTRIBUTE  ::=  {
     WITH SYNTAX BIT STRING
     EQUALITY MATCHING RULE bitStringMatch
     ID id-altSignatureValue }

END
```

Appendix B. Upgrading PKI and Dependent Systems

One way to upgrade these systems is to employ a "top down" approach: First the root CA is upgraded, then the same is done for any subordinate CAs, and finally for end entities. The dependent applications can then be upgraded in phases, where the upgraded applications can switch to using the new public-key algorithms while non-upgraded systems can continue using the old public-key algorithms.

Appendix C. Options for Alternative Algorithms

Out of all branches of mathematics thought to be suitable for quantum-safe cryptographic algorithm development, the theory of hash functions, specifically hash-based signatures are currently the most trusted in regard to their quantum security assurances. While the private key state management makes using them challenging in some high-frequency use cases, they are very well suited for roots of trust and code signing; hash-based algorithms can already be used to upgrade CA certificates. Furthermore, the option will be available to use stateless digital signatures in end-entity certificates when they become available.

Authors' Addresses

Alexander Truskovsky
ISARA Corporation
560 Westmount Rd N
Waterloo Ontario N2L 0A9
Canada

Email: alexander.truskovsky@isara.com

Daniel Van Geest
ISARA Corporation
560 Westmount Rd N
Waterloo Ontario N2L 0A9
Canada

Email: daniel.vangeest@isara.com

Scott Fluhrer
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
United States of America

Email: sfluhrer@cisco.com

Panos Kampanakis
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
United States of America

Email: pkampana@cisco.com

Mike Ounsworth
Entrust Datacard, Ltd
1000 Innovation Drive

Kanata Ontario K2K 3E7
Canada

Email: mike.ounsworth@entrustdatacard.com

Serge Mister
Entrust Datacard, Ltd
1000 Innovation Drive
Kanata Ontario K2K 3E7
Canada

Email: serge.mister@entrustdatacard.com