LAMPS
Internet-Draft
Intended status: Standards Track
Expires: 24 October 2025

S. Sun
Y. He
H. Y. Lin
Huawei

22 April 2025

Convertible Forms with Multiple Keys and Signatures For Use In Internet X.509 Certificates
draft-sun-lamps-hybrid-scheme-01

Abstract

This document presents a hybrid key and signature solution, which allows to integrate two public keys and/or two signatures into a single certificate. The scheme enables a single certificate to be converted between different forms, allowing an alternative public key and/or an alternative signature to be transmitted either by direct inclusion or by referencing external data. This flexibility ensures that the scheme is backward-compatible with legacy devices, while also providing enhanced security support for upgraded devices. Four CSR attributes and two new X.509v3 certificate extensions are defined, and the procedures for signing and verifying certificates containing the defined attributes and extensions are described.

Status of This Memo

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

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

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

This Internet-Draft will expire on 24 October 2025.

Copyright Notice

Table of Contents

1. Introduction

The advent of quantum computing poses a significant threat to the security of traditional cryptographic systems. Classical cryptographic algorithms, such as RSA, ECDSA, and their elliptic curve variants, rely on the computational difficulty of certain number-theoretic problems. However, these algorithms are vulnerable to attacks by adversaries with access to a Cryptograpically-Relevant Quantum Computer (CRQC), which can efficiently solve these problems and compromise the security of the entire system.

To address this vulnerability, there is a pressing need to migrate to post-quantum cryptography (PQC), which employs algorithms resistant to quantum attacks. This transition presents unique challenges, as the security of PQC algorithms is not yet fully established and requires extensive evaluation. Additionally, during the migration period, systems must maintain compatibility with legacy devices that do not support PQC.

This document introduces new Certificate Signing Request (CSR) attributes and X.509v3 certificate extensions to facilitate the migration of post-quantum signature algorithms while ensuring compatibility with existing systems. A key design requirement for this scheme is to enable the coexistence of two public keys and two signatures within a single certificate, and allowing for switching between four distinct certificate forms during transmission without impacting verification.

By embedding an alternative public key and an alternative signature into extensions, multiple certificate chains can be effectively embedded within a single chain. This enables legacy devices to continue using traditional cryptographic algorithms, while upgraded devices can utilize PQC algorithms and remain interoperable with legacy devices. The scheme also allows for the flexible selection of certificate forms, including both "by reference" and "by value" options, to accommodate different device scenarios and optimize bandwidth usage.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

Alternative Subject Public Key: The keys, whose usage is profiled in this document, which can be used to verify a signature instead of, or in addition to, the traditional 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 traditional signature.

Subscriber: An entity that applies a signed certificate.

Relying Party: An entity that receives and verifies certificates.

Issuer: An entity that signs and issues certificates;

Upgraded Device: An entity which is capable of understanding and using the extensions introduced in this document.

2. Design Overview

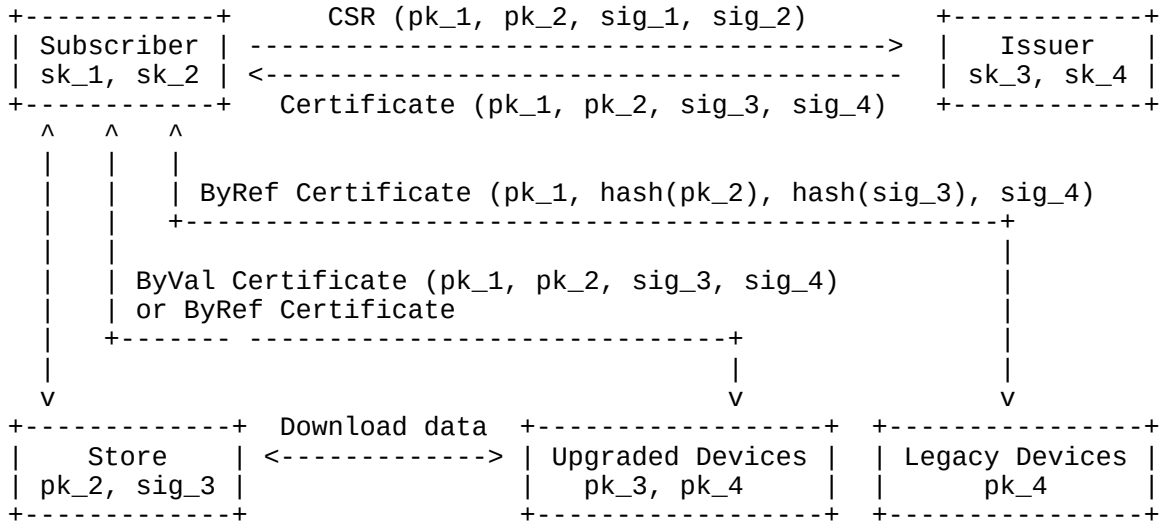

Figure 1: Design Overview

To achieve hybrid signatures in a PKI system, a subscriber needs to apply a certificate that contains two public keys to use hybrid signature schemes. Meanwhile, the issuer needs to sign a certificate with two secret keys and include two signatures in the certificate.

The design of this document is outlined as follows:

1. A subscriber selects two signature algorithms and generates two key pairs (pk_1, sk_1) and (pk_2, sk_2). It then uses sk_1 and sk_2 to generate sig_1 and sig_2, respectively. The signatures are proof-of-possession of the secret keys sk_1 and sk_2. The two public keys and signatures are included in a Certificate Signing Request as defined in Section 4.1.

2. An issuer holds two key pairs (pk_3, sk_3) and (pk_4, sk_4). Upon receiving a CSR, the issuer uses its two secret keys to sign the information provided in the CSR and constructs a certificate as defined in Section 4.3. A certificate has at most four forms as defined in Table 1. The issuer signs Form 4 of the certificate, which includes the values of pk_1 and sig_4, and the hashes of pk_2 and sig_3. The issuer converts the certificate to Form 1 as in Section 4.4, which includes all plain values, and sends the certificate in Form 1 to the subscriber.

3. Upon receiving the certificate, a subscriber puts the alternative public key pk_2 and the alternative signature sig_3 into specific locaitons.

4. For legacy devices that have not been upgraded, only traditional algorithms are used to establish communication with the subscriber. In this case, certificates in Form 4 are transfered, which only include pk_1, sig_4 and hashes of pk_2 and sig_3.

5. For upgraded devices, both traditional algorithms and post-quantum algorithms can be used. In this case, the devices can select a proper certificate form base on their preferences.

3. Alternative Public-Key Algorithm Objects

3.1. OIDs

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

```
id-altSubPubKeyHashAlgAttr   OBJECT IDENTIFIER ::= {TBD}
id-altSubPubKeyLocAttr       OBJECT IDENTIFIER ::= {TBD}
id-altSigValueHashAlgAttr    OBJECT IDENTIFIER ::= {TBD}
id-altSigValueLocAttr        OBJECT IDENTIFIER ::= {TBD}

id-altSubPubKeyExt           OBJECT IDENTIFIER ::= {TBD}
id-altSignatureExt           OBJECT IDENTIFIER ::= {TBD}
```

3.2. CSR Attributes

Four new CSR attributes are defined. The first two attributes are used to submit associated information about the alternative public key of the subscriber for certification. The latter two attributes specify parameters related to the alternative signature generated by the issuer. A CSR MAY include any one or multiple of these attributes.

3.2.1. Alternative Subject Public Key Hash Algorithm Attribute

This attribute specifies the identifier for the algorithm used to hash the alternative public key of a subscriber. The AlgorithmIdentifier type is defined in Section 4.2 of [RFC2986].

This attribute is identified using the id-altSubPubKeyHashAlgAttr OID.

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

3.2.2. Alternative Subject Public Key Location Attribute

This attribute specifies a URI indicating the location where the alternative public key will be stored. This attribute MAY be omitted, if the subscriber and the relying partiy wish to establish a mutually agreed-upon location through external means. The details of this negotiation are outside the scope of this document.

This attribute is identified using the id-altSubPubKeyLocAttr OID.

```
altSubPubKeyLocAttr ATTRIBUTE ::= {
  location     URI
  ID           id-altSubPubKeyLocAttr }
```

3.2.3. Alternative Signature Value Hash Algorithm Attribute

This attribute specifies the identifier for the algorithm used to hash the alternative signature generated by an issuer.

This attribute is identified using the id-altSigValueHashAlgAttr OID.

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

3.2.4. Alternative Signature Value Location Attribute

This attribute specifies a URI indicating the location where the alternative signature generated by the issuer can be stored. The subscriber will store the alternative signature once received from the issuer. This attribute MAY be omitted, if the subscriber and the relying party wish to establish a mutually agreed-upon location through external means. The details of this negotiation are outside the scope of this document.

This attribute is identified using the id-altSigValueLocAttr OID.

```
altSigValueLocAttr ATTRIBUTE ::= {
  location     URI
  ID           id-altSigValueLocAttr }
```

3.3. X.509v3 Extension

Two new X.509v3 extensions are defined. One is designed to include a subscriber's alternative public key in the certificate, while the other is designed to include an issuer's alternative signature. Both extensions have a similar structure.

3.3.1. Alternative Subject Public Key Extension

This extension is identified using the id-altSubPubKeyExt OID.

Conforming issuers MUST mark this extension as non-critical.

This extension, containing the alternative public key and associated metadata, is the DER encoding of the following structure:

```
AltSubPubKeyExt ::= SEQUENCE {
  byVal BOOLEAN DEFAULT FALSE,
  plainOrHash BIT STRING,
  altAlgorithm AlgorithmIdentifier,
  hashAlg AlgorithmIdentifier OPTIONAL,
  location URI OPTIONAL }
```

The fields of extension AltSubPubKeyExt have the following meanings:

* byVal is a boolean with a default value of FALSE. This field indicates whether the alternative public key is transferred by its actual value or by reference, and determines the content of the plainOrHash field.

* plainOrHash stores the alternative public key or its hash value. When byVal is FALSE, the plainOrHash field stores the hash value of the alternative public key. When byVal is TRUE, plainOrHash stores the actual value of alternative public key.

* altAlgorithm identifies the algorithm of the alternative public key.

* hashAlg and location fields are optional. hashAlg field indicates the algorithm used to hash the alternative public key and location field represents the location where the actual value of alternative public key is stored.

3.3.2. Alternative Signature Extension

This extension is identified by the id-altSignatureExt OID.

Conforming issuers MUST mark this extension as non-critical.

This extension contains the alternative signature generated by the issuer, which is the DER encoding of the following structure:

```
AltSignatureExt ::= SEQUENCE {
  byVal BOOLEAN DEFAULT FALSE,
  plainOrHash BIT STRING,
  altSigAlgorithm AlgorithmIdentifier,
  hashAlg AlgorithmIdentifier OPTIONAL,
  location URI OPTIONAL }
```

The fields of the AltSignatureExt extension have the following meanings:

* byVal is a boolean with a default value of FALSE. This field indicates whether the alternative signature is transferred by its actual value or by reference, and determines the content of the plainOrHash field.

* plainOrHash stores the alternative signature or its hash value. When byVal is FALSE, the plainOrHash field stores the hash value of the alternative signature. When byVal is TRUE, plainOrHash stores the actual value of signature.

* altSigAlgorithm identifies the algorithm to generate alternative signature.

* hashAlg and location fields are optional. hashAlg field indicates the algorithm used to hash the alternative signature and location field represents the location where the actual value of alternative signature is stored.

4. Scheme Workflow: From CSR to Signed Certificate

4.1. Creating CSRs

A Certificate Signing Request (CSR) has three fields, as defined in Section 4.2 of [RFC2986].

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

The signature is the result of signing the ASN.1 DER encoding of the certificationRequestInfo field with a subcriber’s private key.

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

The syntax of CSR allows a subscriber to proof possession of one secret key. To prove possession of two secret keys, there are many possible ways. This document adopts the composite signature schemes as defined in a recent IETF draft [I-D.draft-ietf-lamps-pq-composite-sigs]. Namely, two public keys are concatenated to obtain a single public key, and two signatures are concatenated to obtain a single signature. So the syntax of CSR does not need to be changed. In the following description, the terms CompositeKeyGen, CompositeSign, and CompositeVerify refer to the KeyGen, Sign, and Verify algorithms defined in Section 4 of[I-D.draft-ietf-lamps-pq-composite-sigs], respectively.

At the same time, the subscriber is allowed to select one hash algorithm and a location for the alternative subject public key, and another hash algorithm and location pair for the alternative signature generated by the issuer. These four fields are included as additional CSR attributes.

The process of generating a CSR is detailed as follows:

1. Select two signature algorithms A1 and A2 and calls CompositeKeyGen algorithms to generate a key pair (pk, sk). Internally, pk consists of pk_1 and pk_2. sk consists of sk_1 and sk_2. Some combinations and their identifiers have already been defined in Section 7. of [I-D.draft-ietf-lamps-pq-composite-sigs]. Specially, pk_2 is the alternative public key.

2. Construct a CompositeSignaturePublicKey object from the pk as defined in Section 5.2. of [I-D.draft-ietf-lamps-pq-composite-sigs].

   CompositeSignaturePublicKey ::= SEQUENCE SIZE (2) OF BIT STRING

3. Construct a SubjectPublicKeyInfo object from the CompositeSignaturePublicKey object and the algorithm identifier of the selected composite signature scheme.

4. Select two hash algorithms: one for hashing alternative public key pk_2, specified in the altSubPubKeyHashAlgAttr attribute, and another for hashing alternative signature, specified in the altSigValueHashAlgAttr attribute. The subscriber MAY leave these two attributes empty, in which case the hash algorithm specified in the selected composite signature scheme will be used.

5. Select two locations: one for storing the plain alternative public key pk_2, specified in altSubPubKeyLocAttr attribute, and another one for storing the alternative signature, specified in altSigValueLocAttr attribute. The subscriber MAY leave these two attributes empty.

6. Construct a CertificationRequestInfo object from the constructed SubjectPublicKeyInfo object, the four new CSR attributes (AltPublicKeyHashAlgorithmAttr, altSigValueHashAlgAttr, altSubPubKeyLocAttr, and altSigValueLocAttr), and other attributes.

7. Sign the CertificationRequestInfo object with the algorithm CompositeSign using the private key sk.

8. Construct a CertificationRequest object from the CertificationRequestInfo object, the identifier of the compsite signature scheme, and the signature.

4.2. Verifying CSRs

An issuer verifies a CSR by verifying the composite signature using the algorithm CompositeVerify with the pk extracted from CSR.

4.3. Creating 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 }
```

The tbsCertificate field contains the subject and issuer names, a public key associated with the subject, a validity period, and other associated information and extensions.

To include two subject public keys in a certificate, the AltSubPubKeyExt extension is used to hold the alternative public key.

Meanwhile, if an issuer wants to use two different cryptographic algorithms to sign a certificate, the AltSignatureExt extension is used to hold the alternative signature.

To facilitate understanding, a preTbsCertificate is defined, which has the same type as TBSCertificate:

```
preTbsCertificate ::= TBSCertificate
```

The preTbsCertificate is signed by the alternative private key to generate the alternative signature. This alternative signature is then used to construct the AltSignatureExt extension, which is appended to the end of the extension list in the preTbsCertificate, resulting the tbsCertificate object. In other words, tbsCertificate is essentially preTbsCertificate appended with the AltSignatureExt extension.

After verifying a CSR, the issuer proceeds with the certificate creation process, which can be divided into 3 stages:

1. Creating AltSubPubKeyExt extension and preTbsCertificate.

2. Creating AltSignatureExt extension and tbsCertificate.

3. Creating certificates.

4.3.1. Creating AltSubPubKeyExt extension and preTbsCertificate

To create an AltSubPubKeyExt extension, an issuer does the following steps:

1. Extract the two public keys pk_1 and pk_2 from the pk in CSR. Obtain the two component algorithm identifiers from the composite signature scheme as defined in Table 2. of [I-D.draft-ietf-lamps-pq-composite-sigs].

2. Construct a SubjectPublicKeyInfo object from pk_1 and the algorithm identifier of the first public key.

3. Determinate the hash algorithm to be used to generate a hash of the alternative public key by checking the AltPublicKeyHashAlgorithmAttr attribute in the CSR. If this attribute is not present, use the hash algorithm defined in the composite signature scheme.

4. Compute a hash by hashing pk_2 with the determined hash algorithm.

5. Construct an AltSubPubKeyExt extension: a) Set the byVal field to FALSE. b) Set the plainOrHash field to the hash calculated in step 4. c) Set the altAlgorithm field to the algorithm identifier of the alternative public key. d) Set the hashAlg to the identifier of hash algorithm determinated by step 3. e) Set the location field to the location field given in the altSubPubKeyLocAttr attribute in the CSR. If this attribute is not present, leave the location field empty.

After creating an AltSubPubKeyExt extension, an issuer constructs a TBSCertificate object from attributes in the given CSR following existing standards, e.g., [RFC2986] and [RFC5280]. The constructed TBSCertificate object is the preTbsCertificate field, which MUST inlude the created AltSubPubKeyExt extension.

4.3.2. Creating AltSignatureExt extension and tbsCertificate

To create an AltSignatureExt extension, an issuer does the following steps:

1. Determine the hash algorithm to be used to generate a hash of the alternative signature by checking the altSigValueHashAlgAttr attribute in the given CSR. If this attribute is not present, use the hash algorithm defined in the composite signature scheme.

2. Sign the preTbsCertificate constructed in Section 4.3.1 with the issuer’s alternative private key to obtain the alternative signature.

3. Compute a hash by hashing the alternative signature (obtained in step 2) with the hash algorithm determined in step 1.

4. Construct an AltSignatureExt extension: a) Set the byVal field to FALSE. b) Set the plainOrHash field to the hash calculated in step 3. c) Set the altAlgorithm field to the algorithm identifier of the alternative signature. d) Set the hashAlg field to the identifier of the hash algorithm determined in step 1. e) Set the location field to the location field given in the altSigValueLocAttr attribute in the CSR. If this attribute is not present, leave the location field empty.

After creating an AltSignatureExt extension, an issuer obtains a new TBSCertificate object by appending the created extension to the extension list of preTbsCertificate.

4.3.3. Creating Certificate

An issuer signs the tbsCertificate with its first private key and algorithm, and then constructs the digital certificate as defined in [RFC5280].

Note that the certificate has both the AltSubPubKeyExt and AltSignatureExt extensions with byVal fields set to FALSE, indicating that the certificate is signed with the hashes of the alternative public key and the alternative signature.

4.4. Converting Forms of Certificates

4.4.1. Converting Forms of Extensions

The two newly introduced extensions, AltSubPubKeyExt and AltSignatureExt, are independent and each can exist in one of two forms:

* ByValue: The extension provides the actual value of a public key or a signature. Namely, the plainOrHash field is set to the actual value, and the byVal field is set to TRUE.

* ByReference: The extension provides the hash of a public key or a signature. Namely, the plainOrHash field is set to the hash of the plain value, and the byVal field is set to FALSE.

The exact form of a given extension object can be determined by checking the byVal field.

To convert a ByValue form to a ByReference form, an entity sets the byVal field to FALSE, computes the hash of the plainOrHash field, and replaces the plainOrHash field with its hash.

To convert a ByReference form to a ByValue form, an entity sets the byVal field to TRUE, replaces the plainOrHash field with the actual value. The entity MUST have the actual value of the alternative public key or the alternative signature.

Note that the other fields in the extensions remain the same in both forms.

4.4.2. Converting Forms of Certificates

A certificate contains the two proposed extensions can exist in 4 forms as follows.

	AltSubKeyValueExt	AltSignatureExt
Form 1	ByValue	ByValue
Form 2	ByValue	ByReference
Form 3	ByReference	ByValue
Form 4	ByReference	ByReference

Table 1: Form of Certificate

To convert a certificate from one form to another form, an entity converts the included extensions from one form to another form following the Section 4.4.1.

During the issuance of a certificate, Form 4 is the signed form. After constructing a certificate as in Section 4.3, an issuer MUST send the certificate in FORM 1 or FORM 3 to the subscriber. Otherwise, there is no way for the subscriber to obtain the value of the alternative signature.

During the communication of a subscriber and a relying party, all of these forms MAY be used to transfer a certificate. However, all other forms MUST be converted to Form 4 before certificate verification. Meanwhile, an entity MAY fetch the alternative public key and/or the alternative signature from the given locations. The entity MAY also cache the alternative materials locally. Figure 2 shows the example process:

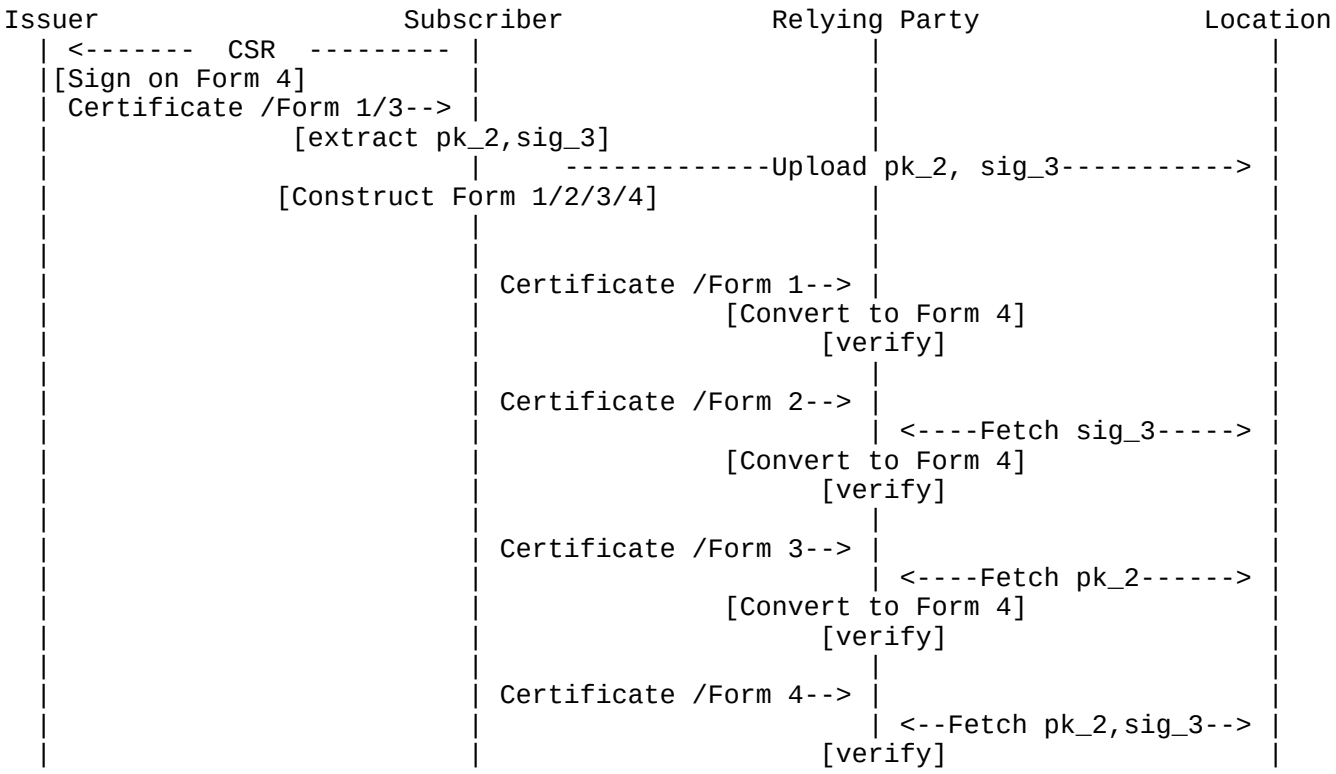

Figure 2: Transmission and Verification of Different Certificate Forms

4.5. Verifying Certificates

Given a certificate, a relying party MAY download the actual value of a public key or a signature if a extension is given in ByReference form.

1. Retrieve the external reference address from the location field in the extension, and fetch the actual value from that address. If the location attribute is absent, the actual value can be obtained through other agreed-upon means, i.e., from previous local cache. If the actual value cannot be obtained, this procedure fails.

2. Determine the hash algorithm by checking the hashAlg field in the extension.

3. Compute the hash of the actual value and compare it with the plainOrHash field. If they do not match, this procedure fails.

To verify a certificate, a relying party MUST convert it into FORM 4 as described in Section 4.4. After the conversion, the relying party verifies the signature field using the issuer's public key following [RFC5280]. Meanwhile, the relying party MAY construct the PreTBSCertificate object by removing the AltSignatureExt from the extension list of the TBSCertificate object, and verify the altSignatureValue field using the issuer's alternative public key.

4.6. Revoking Certificates

In certain situations, certificates must be revoked. This can occur due to various reasons, such as key compromise, CA compromise, or changes in affiliation. Generally, Certificate Revocation Lists (CRLs) are authenticated lists of revoked certificates, as defined in [RFC5280].

In the context of certificates with multiple keys and signatures, the compromise of any private key (whether traditional or alternative) MUST result in the revocation of the entire certificate. Conversely, if a certificate with multiple keys and signatures is revoked, both the traditional and alternative keys SHOULD be treated as revoked. This approach prevents the unnecessary complexity of managing a certificate where one key is compromised while the other remains secure.

5. Use Case: Post-Quantum Migration

During the post-quantum migration, there is going to be a long period of time where legacy devices and upgraded devices coexist. In this scenario, when applying for a certificate, the subscriber selects a traditional signature algorithm for the first public key and a post-quantum signature algorithm for the alternative public key. The issuer also uses a traditional signature algorithm and a post-quantum signature algorithm for its first and alternative public keys, respectively.

Assume that a device have some mechanisms to share its capabilities with a subscriber. For example, in TLS, the client sends a ClientHello message, which includes its supported algorithm list and a set of extension requests.

Legacy devices only support traditional algorithms and do not recognize the AltSubPubKeyExt extension. In this case, the subscriber sends a certificate in Form 4 to legacy devices. Then, the legacy device ignores the proposed two extensions, verifies the certificate with the issuer's traditional public key, and retrieves the subject's traditional public key from the certificate. The legacy device can then use the subject's traditional public key to

verify the identity of the subscriber. This allows legacy devices to continue using traditional algorithms as before, ensuring backward compatibility.

Upgraded devices, which support both traditional and post-quantum algorithms, recognize the proposed two extensions. An upgraded device can request any form of a certificate according to its preference. Given a certificate, the device determines its form and verifies the certificate follows Section 4.5. After the verification, the device can use subject's one or two public keys to verify the identity of the subscriber.

An upgraded device can obtain the actual value of the alternative materials by requesting a certificate in FORM 1 during the first communication with the subscriber and cache the alternative public key. Later the device can select certificates in other forms to save transmission overhead.

6. IANA Considerations

TODO: ADD a table to include the required OIDs.

7. Security Considerations

Many of the security considerations for this document closely follow those of [RFC5280] and [I-D.draft-truskovsky-lamps-pq-hybrid-x509]. However, the extension introduced in this document does bring rise to additional considerations.

The certificate issued as in this document MAY has location fields to allow the relying party to download the alternative materials. It is possible that a malicious actor replaces the content at the refered location, but this action will be detected since the hash of the actual value is given in the certificate.

This document uses the composite hybrid signature scheme in CSR follows the description in [I-D.draft-truskovsky-lamps-pq-hybrid-x509]. It achieves the Non-Separability security property, which is defined in Section 1.4.3. of [I-D.draft-ietf-pquip-hybrid-signature-spectrums]. So an attempt to remove any of the public key or signature of a subscriber will be detected.

For the certificate issuance, the issuer uses the nested hybrid signature scheme to sign the information of a certificate with two private keys. To achieve backward-compatibility, the subscriber SHOULD use a traditional signature algorithm and a post-quantum signature algorithm for the outer and inner signature algorithms,

respectively. One of the main drawback of nested hybrid signature is that the outer signature can be stripped. But in the case of a certificate, the strpping attack will be detected. Another problem is that if a malicious actor with a CRQC breaks the outer traditional signature algorithm, it can alter all the inner message, including the location, flag and hash for the inner signature. In this powerful attack, a relying party will download a forged signature from a location controlled by the attacker and passed the hash comparison. But when the relying party use the issue's alternative public key to verify the downloaded signature, the verification will certainly fail.

For certificate transparency services, certificates in FORM 4 SHOULD be logged and monitored.

For a legacy device, to achieve backward-compatibility, it uses the signature field to verify the certificate and the first public key of the subscriber to authenticate handshake transcripts. In this case, there is no post-quantum security.

For an upgraded device, the situation is more complicated. It can only use the first public key or two public keys. It can only verify the signature field or verify two signatures. All these choices depend on the communication parties' preferences and/or configuration. The mechanism described in this document gives 4 forms of a certificate. However, the upper-layer applications or protocols needs to decide the strategy when applying the proposed mechanism.

8. References

8.1. Normative References

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[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/rfc/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/rfc/rfc5280>.

[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/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

8.2. Informative References

[I-D.draft-truskovsky-lamps-pq-hybrid-x509] Truskovsky, A., Van Geest, D., Fluhrer, S., Kampanakis, P., Ounsworth, M., and S. Mister, "Multiple Public-Key Algorithm X.509 Certificates", Work in Progress, Internet-Draft, draft-truskovsky-lamps-pq-hybrid-x509-02, 24 August 2023, <https://datatracker.ietf.org/doc/html/draft-truskovsky-lamps-pq-hybrid-x509-02>.

[I-D.draft-ietf-lamps-pq-composite-sigs] Ounsworth, M., Gray, J., Pala, M., Klauテ殤er, J., and S. Fluhrer, "Composite ML-DSA for use in X.509 Public Key Infrastructure and CMS", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-sigs-04, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-04>.

[I-D.draft-ounsworth-lamps-pq-external-pubkeys] Ounsworth, M., Gray, J., Hook, D., and M. O. Saarinen, "External Keys For Use In Internet X.509 Certificates", Work in Progress, Internet-Draft, draft-ounsworth-lamps-pq-external-pubkeys-05, 8 October 2024, <https://datatracker.ietf.org/doc/html/draft-ounsworth-lamps-pq-external-pubkeys-05>.

[I-D.draft-ietf-pquip-hybrid-signature-spectrums] Bindel, N., Hale, B., Connolly, D., and F. D, "Hybrid signature spectrums", Work in Progress, Internet-Draft, draft-ietf-pquip-hybrid-signature-spectrums-06, 9 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-hybrid-signature-spectrums-06>.

Appendix A. Appendix 1 [REPLACE/DELETE]

This becomes an Appendix [REPLACE]

Acknowledgements

This template uses extracts from templates written by Pekka Savola, Elwyn Davies and Henrik Levkowetz. [REPLACE]

Authors' Addresses

Shuzhou Sun
Huawei
China
Email: sunshuzhou@huawei.com

Yidi He
Huawei
Singapore
Email: heyidi@h-partners.com

Hsiao Ying Lin
Huawei
France
Email: lin.hsiao.ying@huawei.com