Post-quantum Multi-Key Mechanisms for PKIX-like protocols; Problem Statement and Overview of Solution Space
draft-pq-pkix-problem-statement-01

Abstract

With the widespread adoption of post-quantum (PQ) cryptography will come uncertainty about the strength of cryptographic primitives. For example; when will RSA and ECC fall? Are Lattice schemes as strong as we believe? The cryptographic community is calling for hybrid schemes that combine classic and post-quantum crypto in ways that remain strong so long as at least one of the algorithms used remains strong.

This document defines the problem statement for digital signatures in PKIX-like protocols, and gives an overview of the general families of solutions.

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 March 20, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
In general terms, "hybrid" or "layered" signatures means that the document signer performs multiple, parallel, signatures over the document and provides them all to the verifier. The verifier’s job is to check that all signatures are valid.

The general concept is straight-forward, but the devil is in the details.

1.1. Formal Security Requirements

A solution to the PQ multi-signature problem MUST meet the following "security" properties:

- S1. In order to break the overall signature, an attacker must break all the signatures.

- S2. Robust to stripping and substitution attacks, which in most cases reduces to the following statement: the verifier needs a way to know which and how many signature algorithms to expect on a given message.

A solution to the PQ multi-signature problem MAY meet the following "desirable" properties, depending on context of the protocol in which
the signature is being performed. And yes, some of these conflict with each other:

- **D1.** The set of signing algorithms is negotiated dynamically between client and server.

- **D2.** Since post-quantum public keys and signatures are large, they should only be sent when you know the client will verify them. For bonus points, the protocol should control when and how the large PQ data is transmitted. Note that all three schemes below could include a hash and external delivery of the large post-quantum data (for example, HTTP URL, delivered via protocol extension, etc). Since this trick is orthogonal to and applies equally to all three schemes, it is not considered here.

- **D3.** Backwards compatibility: there is a mechanism either for the client to negotiate whether it wants multi-key signatures, or the signature mechanism is designed in such a way that a legacy verifier will ignore the PQ parts.

- **D4.** Protocol compatibility: a multi-key solution should "drop-in" to existing protocol message formats. For example, a solution where any PKIX-like protocol simply needs to pick up new OIDs with no other code changes would meet this property. Note that this conflicts with D2 which (probably) requires protocol-level awareness of the multiple keys.

- **D5.** A mechanism that supports 3 or more algorithms is desirable to hedge our bets against algorithm compromise; possibly allowing a set of public keys / signatures to continue being used so long as there remain an acceptable number of un-broken algorithms.

- **D6.** Some applications may be limited to a single signing certificate and/or key without significant redesign (for example smart cards).

### 2. Solution Space

At the time of writing, there have been proposed three broad families of solutions being considered: concatenating multiple public keys and signatures together into a large single public key / signature object, multiple independent traditional certificate chains, and placing PQ data into v3 extensions. Since this discussion came out of LAMPS, the solutions that have drafts are X.509-focused, but we note that the first solution ("composite") is more general.

Each solution space is described below. I am trying to keep these abstract and not solution-specific.
2.1. "Composite" concatenated keys and signatures

Concatenate public keys, algorithmIDs, and signatureValues into "composite" versions of those structures.

Pros:

- D4: "Drop-in" to most protocols because once the underlying crypto layer supports the composite primitive, then the protocol only needs to add support for the composite algorithmID.

- D5: Trivially extends to 3 or more algorithms.

- Complexity of composite signature generation and verification is moved to crypto layer, and therefore protocol designers and implementers do not need to worry about it. Note this is also a con, see below. Crypto layer maintainers have control of which combinations of algorithms are acceptable. This could be exposed to the application layer, for example, through config files or APIs.

- D6: Single certificate and key object should be a near drop-in replacement for applications such as smart cards whose architectures limit them to a single certificate.

Cons:

- D1: Does not apply well to negotiated protocols, at least not without a quadratic or exponential proliferation of certificates using every combination of algorithms (possibly equip servers with two certificates: a high-security certificate, and a high-compatibility one). Another option is to allow servers to perform signatures with a subset of keys in the certificate, or allow clients to selectively ignore some signatures. Note that this requires a complex verification algorithm which is easy to mis-implement, and makes it very difficult to detect stripping attacks. So this is probably not a good direction to go.

- D2: All public keys and signatures are contained in the certificate, and therefore need to be transferred.

- D3: Is not immediately backwards-compatible since clients need to be patched to understand the composite message structures and OID. But once that’s done the verification algorithm can be designed to skip unknown algorithmIDs (this is not unique to this scheme, and violates D2, but it’s possible _shrug_).
Policy moved to crypto layer; protocol can not (easily) be negotiate the algorithms, or control what counts as an "acceptable" combination of algorithms.

Neutral:

S1 & S2: Substitution and stripping attacks; if an attacker can break one or more of the signature algorithms, then they can strip out the other algorithmIDs and re-generate the signatures they have broken. This can be mitigated by requiring that a composite public key produces signatures using all keys. A verifier can check against their local trust store to see that the EndEntity certificate contains the same algorithms as its issuer.

D2: Large PQ data is always sent to the client. This can be either viewed as a bad thing (bandwidth) or a good thing (protect data now, patch clients later).

This family of solutions has been instantiated in [draft-ounsworth-pq-composite-sigs-01].

2.2. Application: X.509

2.2.1. Multiple Certificate Chains

If you want multiple public keys on multiple cryptographic algorithms, then get multiple certificates from multiple PKIs. The protocol has control of negotiating which algorithms get used, how to encapsulate the large PQ data, etc. In many ways, this is the most obvious solution.

Pros:

D1: Highly friendly to negotiated protocols since it allows the server to sign a transaction with the combination of algorithms that was negotiated with the client.

D2: Protocol has control of how and when to transmit the large PQ certificates and signatures.

During the RSA to ECC migration, many web servers added support for loading multiple certificates, and deciding which to use based on the TLS negotiation, so this work is already half done.

D3: Clients will only be served certificate algorithms that they have requested.
o D4: Does not require any modification to PKI hierarchies. Does not require any changes to X.509, and only minor changes to CMS (for example, current CMS implementations have inconsistent behaviour when multiple SignerInfos are present).

o D5: Trivially extends to 3 or more certificates on different algorithms.

Cons:

o S1 & S2: Unclear how this applies to non-negotiated protocols.

o S1 & S2: mitigation against stripping and substitution attacks is left up to the protocol, and is easy to get wrong. Possibly some kind of extension could be placed in root CA certs to indicate which "sister PKIs" a verifier can expect to see, but such a solution would be fairly complex.

o D4: Requires updates to all protocols to add or specify the behaviour of multiple SignatureAlgorithm and SignatureValue fields.

o D6: Will have compatibility issues with applications such as smart cards that are limited to a single signing certificate.

At time of writing, I am not aware of any internet drafts or other implementations of this family of solutions.

2.2.2. "Hybrid" v3 extensions

Place the PQ data into X.509v3 extensions. This has two general forms; the "PQ extension" contains a complete second certificate, or the "PQ extensions" contain an alternative public key, signature algorithm, and signature value.

Pros:

o D3: Highly backwards compatible; if the PQ extension(s) are marked as non-critical, then legacy clients will ignore the PQ data and verify the "outer" signature as normal.

o D6: Single certificate should be compatible with applications such as smart cards whose architectures limit them to a single certificate. Though API / communication with the card would need to support two signatures or challenge-responses.

Cons:
o D1: Does not apply well to negotiated protocols; at least not without a quadratic proliferation of certificates on every combination of algorithms.

o D3: May be too backwards-compatible; in a large PKI deployment, it is very hard to be know if/when clients are all using the PQ data.

o D4: Requires updates to all protocols to add or specify the behaviour of multiple SignatureAlgorithm and SignatureValue fields.

o D5: Does not extend to 3 or more keys + signatures on its own, but could be used in combination with the "composite" solution to do so.

Neutral:

o S1 & S2: Substitution and stripping attacks; if an attacker can break the outer signature (assumed to be RSA for backwards compatibility reasons), then they can strip or substitute the PQ extension and re-generate the outer signature. This can be mitigated by requiring that a hybrid CA containing an alt public key always produces hybrid certificates containing a corresponding alt signature. When building a certificate chain, a verifier can check that alt signatures are present and valid all the way down the certificate chain.

o D2: Large PQ data is always sent to the client. This can be either viewed as a bad thing (bandwidth) or a good thing (protect data now, patch clients later).

This family of solutions has been instantiated in [draft-truskovsky-lamps-pq-hybrid-x509-01], and has a related standard currently before the ITU.

3. Conclusion

None of these solution families are a panacea. Multi-cert looks preferable for online negotiated protocols, hybrid looks preferable for environments with strong backwards compatibility requirements for legacy clients, and composite looks preferable for controlled environments where you know the clients will support the composite message format.
4. Intellectual Property Considerations

Hybrid certificates, specifically [draft-truskovsky-lamps-pq-hybrid-x509-01] has IPR held by ISARA which has an IPR statement available at https://datatracker.ietf.org/ipr/3287/.

Composite certificates, specifically [draft-ounsworth-pq-composite-sigs-01] has IPR held by Max Pala and CableLabs, but with open license terms, available at https://datatracker.ietf.org/ipr/3481/.

5. Contributors and Acknowledgements

This document incorporates contributions and comments from a large group of experts, including the authors list of [draft-ounsworth-pq-composite-sigs-01] and hallway chats at IETF105 and other crypto conferences.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

6. Informative References

[I-D.ounsworth-pq-composite-sigs]
Ounsworth, M. and M. Pala, "Composite Keys and Signatures For Use In Internet PKI", draft-ounsworth-pq-composite-sigs-01 (work in progress), July 2019.

[I-D.truskovsky-lamps-pq-hybrid-x509]


Author’s Address