And TLS lived happily ever after

Michael Scott, Gora Adj and Francisco Rodríguez-Henríquez
Cryptography Research Centre
Technology Innovation Institute
{michael.scott,gora.adj,francisco.rodriguez}@tii.ae

Abstract

The plausible threat of a Cryptographically Relevant Quantum Computer (CRQC) has rightly stimulated a move away from traditional methods of asymmetric cryptography to new post-quantum secure equivalents. Digital signature is the cryptographic primitive that authenticates an Internet server's identity by signing each certificate in an X.509 certificate chain. A suggested response to the CRQC threat is to deploy a hybrid classical/post-quantum digital signature, combining a traditional tried-and-tested scheme with a post-quantum alternative, where certificates are signed using both methods. Here we propose a fused signature scheme that adopts the same approach, but introduces minimal friction into existing TLS architectures.

1 Introduction

A Cryptographically Relevant Quantum Computer (CRQC) represents a plausible threat to classic methods of asymmetric cryptography. That is plausible, not imminent, not even inevitable. Nevertheless, it is a threat which deserves a response. The cryptographic community has responded quickly with new methods for key exchange and digital signature which are immune to a CRQC, and have already advanced to a stage where such methods are standardised.

TLS is the most important cryptographic protocol in use today. It is activated everytime a website is accessed. It protects and hence makes possible e-commerce on the Internet. You will have activated it in the process of viewing this preprint. Ideally our response to the threat of a CRQC would merely require the replacement of older primitives with newer and improved alternatives, something which indeed has happened in the past as TLS has evolved. However, there are 3 additional problems that can arise in a move to a post-quantum world. (1) The new primitives will have different computational requirements. (2) The new primitives will impact the bandwidth requirements of the protocol, that is the amount of data that needs to be exchanged between server and client during a protocol handshake. (3) The

new primitives will not have yet survived the all-important "test of time", that is their true security in the face of a CRQC, or indeed classical methods of cryptanalysis, is still uncertain.

In the interests of making a TLS implementation as "agile" as possible, adapting to a post-quantum world should simply require the replacement of certain cryptographic primitives, without further deeper structural changes. This would be desirable, but is it possible?

2 Preliminaries

2.1 TLS

TLS version 1.3 is a three-pass protocol that uses a Key Exchange Mechanism (KEM) to establish a mutual secret key, from which are derived keys to encrypt the client-server communication. Until now the X25519 Diffie-Hellman key exchange has been popular, and requires the exchange of only 32 bytes of data. However, already the threat of harvest-now-decrypt-later attacks has motivated a move to a hybrid method. Post-quantum KEMs have been proposed and standardised, the most popular being the lattice-based scheme known as ML-KEM, and the X25519MLKEM768 hybrid is already on its way to becoming a de-facto standard in TLS.

Let us briefly recap how the TLS protocol works in its most common configuration. We consider only the most recent version of TLS, that is version 1.3 [38].

A client sends a bunch of stuff to the server, including an ephemeral KEM public share. The server sends a bunch of stuff back, including its ephemeral KEM public share. This exchange of public shares is used to derive a mutual (but unauthenticated) secret key. The client sends some final stuff before dropping down to an encrypted and one-way authenticated communication between client and server. As early as possible the protocol data flow is encrypted using the secret key, which is possible partway through the server's response (or even earlier if Encrypted Client Hello [39] is enabled).

Typically the client-to-server protocol messages tend to be small. However the server-to-client communication can be quite large, as it may include a PKI (Public Key Infrastructure) certificate chain. This certificate chain performs a vital function – it authenticates the server to the client by securely providing a public key to the client, which allows it to verify the server's signature on the protocol transcript.

It is important to note here that once the server has authenticated itself, it may send the client a resumption ticket, to be stored typically as a browser cookie. This allows the client to resume at a later time (maybe as long as a week later) without the requirement for the certificate chain to be sent again. This greatly reduces the amount of communication required in the handshake (the KEM share values typically remain, but resumption removes

digital signature/verification completely from the handshake). However we will consider here only the case where a certificate chain will be transmitted.

2.2 PKI

All X.509 digital certificates contain details of the entity that owns it and an associated Public Key. Appended is the digital signature of another entity which attests to the certificate's validity. Each certificate entity owns and protects the private key associated with that embedded Public Key.

A TLS certificate chain typically consists of three digital certificates. At the root lies a self-signed Certificate Authority (CA) certificate. The recognised certificate authorities in the world number in the low hundreds. Next is an intermediate authority certificate, signed by the root owner's private key. Finally comes the leaf (or server) certificate, which represents the "business end" of the chain. Its owning entity is typically the identity/URL of the website being visited by the client. The leaf certificate is signed by the intermediate's private key. The public key of the leaf certificate and its associated private key are used as described above by the TLS protocol.

Root certificates, since there are a finite number of them, are commonly stored by the client, and hence do not need to be transmitted during the protocol handshake.

Each certificate has a lifetime. Typically a root certificate might have a lifetime of 5 years, an intermediate certificate of 2 years, and a leaf certificate of 1 year. However there are moves afoot to reduce leaf certificate lifetimes down to as low as 47 days, or even shorter. This has benefits, as it means that (a) the leaf certificate may be justified in supporting a lower security level as an attacker would need to break it within its lifetime, and (b) certificate revocation, a major bugbear of PKI, becomes less of an issue.

2.3 Post-Quantum signatures

Post-quantum signature schemes have already been proposed and standardised after intense scrutiny by the cryptographic community. The effort is ongoing and new schemes can expect to be recommended in the near future. In time they may expect to completely replace the classical RSA and ECC methods. See Table 1

A problem is already evident. Each certificate contains a public key and a digital signature. Using the currently standardised PQ algorithms this means that the certificate chain will increase dramatically in size and the bandwidth requirement of the protocol may start to become an issue. But note our inclusion of the not-currently-standardised isogeny-based scheme SQIsign. Using isogenies we have the potential for much smaller public keys and signatures.

Table 1: Public-key/signature sizes in bytes

Algorithm	Level 1	Level 3	Level 5
NIST-ECC[a][28, 30, 29, 43]	65/64	97/96	133/132
RSA[b][28, 29, 25]	384/384	960/960	1920/1920
ML-DSA [33] (PQ)	1312/2420	1952/3309	2592/4627
FN-DSA [31] (PQ)	897/666	–	1793/1280
SLH-DSA [32] (PQ)	32/7856	48/16224	64/29792
SQIsign (PQ)	**65/148**	**97/224**	**129/292**

[a] Uncompressed EC public keys and fixed-width raw ECDSA signatures.
[b] RSA row reports modulus bytes / signature bytes. A fully encoded RSA public key in X.509/SPKI is slightly larger.

We shall refer to each byte of cryptographic material as a cryptobyte. Assume a PQ implementation of TLS version 1.3 which uses X25519MLKEM768 for key exchange. This requires the server to send a 1120 cryptobyte public share to the client.

Consider a certificate chain that used ML-DSA at security level 1 for the leaf certificate, and ML-DSA level 3 for the intermediate and root certificates. The leaf certificate would contain 1312+3309 cryptobytes. The intermediate certificate would require 1952+3309 cryptobytes. The TLS server transcript signature would require 2420 cryptobytes. Adding in the KEM key share that is a total of 13422 cryptobytes to be transmitted from server to client.

And that is before we even consider X.509 and other protocol overheads. Some experimental evidence suggests that a transmission of anything over 10000 bytes would start causing performance problems for TLS [44], [45]. A solution being pushed by Google is to use so-called Merkle Tree Certificates. See [21] for an unbiased review of the advantages and disadvantages of this approach. However some concern has been expressed that Merkle Tree Certificates as proposed may represent an undesirable concentration of control over the Internet in the hands of browser vendors. And is it strictly necessary?

Consider next a chain that used ML-DSA at security level 1 for the leaf certificate, and SQIsign level 3 for the intermediate and root certificates. The leaf certificate would contain 1312+224 cryptobytes. The intermediate certificate would require 97+224 cryptobytes. The TLS server transcript signature would again require 2420 cryptobytes. Adding in the KEM key share yields a total of 5397 cryptobytes to be transmitted from server to client.

The above analysis excludes the cost of signatures associated with the optional Certificate Transparency mechanism, which would add two more signatures per handshake [44]. This would further significantly impact the

bandwidth requirement for the ML-DSA based solution, but would have much less effect on a SQIsign-based solution.

3 Isogeny-based cryptography

The idea of using isogenies on elliptic curves as a basis for post-quantum cryptography has been around for a long time. But unfortunately it got off to a very bad start. The key exchange algorithm based on isogenies known as SIDH [22] had a surprisingly long history before it was comprehensively broken [10]. Sadly by the time it was broken considerable investments in time and effort had been expended in order to make it practical, which made its fall particularly devastating.

However the isogeny community has picked itself up, and is again quite bullish about the prospects for isogeny-based cryptography [14], [4]. We suspect that it is only a matter of time before a scheme is developed which is competitive with those already standardised. But even now the proposal SQIsign [14] is on the verge of practicality.

One negative characteristic of SQIsign is that signature generation is slow, and hard to make constant time. However, in the scenario considered above SQIsign signing is never required during the execution of TLS, by either client or server. It is only required for the offline construction of the certificate chain. On the other hand SQIsign verification, which is required by the client, is relatively fast (compared with signing) and is easy to implement in constant time.

Many currently proposed schemes like SQIsign use finite field arithmetic based on the primes $5.2^{248}-1$, $65.2^{376}-1$ and $27.2^{500}-1$. Fields of these sizes would be familiar to implementors of standard elliptic curve cryptography, and schemes based on such primes would be expected to hit levels 1,3, and 5 of security in both classic and isogeny-based contexts. These primes p are chosen such that $p+1$ is divisible by a large power of 2, which is important for isogeny-based proposals. However such primes are also Montgomery-friendly [3], and in the past primes of this form were suggested for use in standard classical Elliptic-Curve Cryptography [9].

3.1 SQIsign

The Short Quaternion and Isogeny Signature scheme (SQIsign) currently stands as the leading isogeny-based signature proposal and remains as the only isogeny-based candidate in NIST's call for additional post-quantum signature schemes. NIST launched Round 1 in June 2023 and accepted 40 proposals spanning a broad range of hard problems [34]; in October 2024, it advanced 14 of these schemes to the second round. Among all round 2 candidates, SQIsign distinguishes itself by offering by far the smallest combined signature and public-key size. Its principal drawback, however,

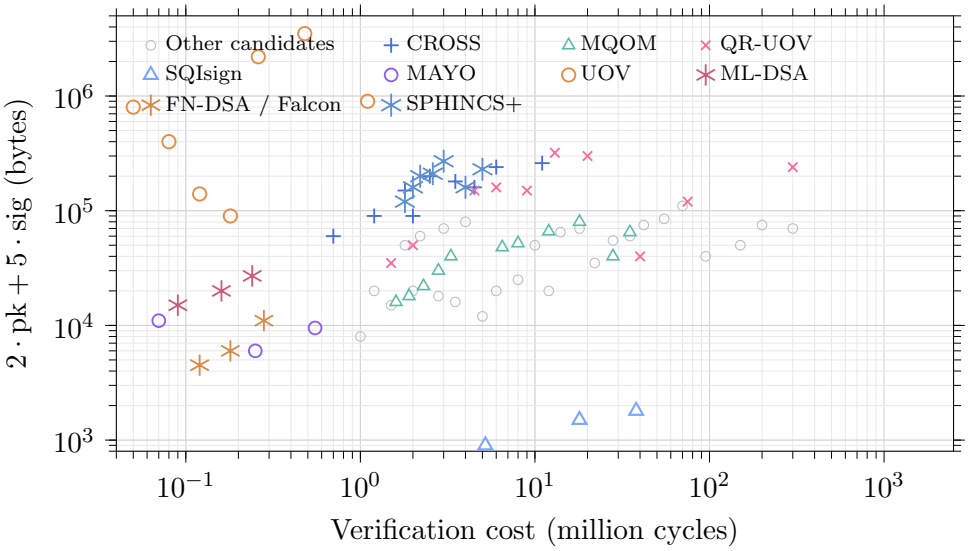


Figure 1: Performance/size trade-off for selected post-quantum signature schemes in a TLS setting. The x-axis reports verification cost (million cycles), and the y-axis reports the metric $2 \cdot \mathrm{pk} + 5 \cdot \mathrm{sig}$ in bytes. SQIsign is shown together with selected comparison families; other candidates are shown in the background (figure adapted from [2]).

is performance: signing remains substantially slower than in lattice-based schemes, verification achieves a decent but moderate efficiency, and current implementations do not operate in constant time [11, 1]. These features make SQIsign especially attractive in applications that prioritize compact signatures and predominantly perform verification, such as certificate infrastructures, software authentication, and resource-constrained devices. Using the PQ-SORT tool [2] we can see in Figure 1 that SQIsign is especially well suited for TLS applications compared to the other NIST additional signature candidates and also to the standardised signature options such as ML-DSA [33], FN-DSA [31] and SLH-DSA [32].

Originally introduced at Asiacrypt 2020 [5], SQIsign has developed rapidly. The first improvements preserved the original framework [15], whereas subsequent work introduced more substantial algorithmic changes. Ironically, a major turning point followed the devastating attacks on SIDH/SIKE [10, 24, 40], which brought higher-dimensional abelian varieties to the forefront of isogeny-based cryptography. Their key tool is Kani's lemma [23], which combines isogenies of degrees a and b into an isogeny of degree $a+b$ between abelian varieties of dimension two or higher. Although this approach increases the dimension of the original isogenies, it can be implemented efficiently by setting $a + b = 2^n$, with n fixed.

This constructive usage of Kani's lemma was first exploited in SQIsignHD [12], which replaced parts of the original protocol with higher-dimensional constructions and reported roughly 40× faster signing together with similarly faster key generation. These gains, however, came at the cost of expensive 4- or 8-dimensional verification. More recently, the algorithmic advances of [26, 37] enabled two-dimensional variants of SQIsign, developed independently in [27, 5, 16]. These schemes use 2D isogenies for both signing and verification, achieving substantial improvements over the original 1D version. Their security analyses also provide a more rigorous foundation and largely remove the heuristic assumptions that appear in the 1D SQIsign presented in [11].

Table 2 highlights the remarkable performance gains of 2D SQIsign [1] over its 1D counterpart [11]. For the former, signature generation and verification now require only 101.6 million and 5.1 million Ice Lake clock cycles, respectively. More recently, the authors of [17] presented a vectorized implementation of SQIsign for several high-performance Arm microarchitectures. Their implementation achieves speedups of 2.24× in key generation and 2.49× in signing relative to the reference implementation of [1]. When combined with the Qlapoti procedure [8] (see [17, Table 6]), SQIsign key generation and signing require only 28.98 million and 73.04 million Cortex-A76 clock cycles, respectively.

Table 2: SQIsign assembly-optimised implementation performance in **millions of clock cycles**

Security level	KeyGen	Sign	Verify
Round 1[*]			
NIST-Level 1	1,661.0	2,370.0	37.0
Round 2[**]			
NIST-Level 1	43.3	101.6	5.1
NIST-Level 3	134.0	309.2	18.6
NIST-Level 5	212.0	507.5	35.7

[*] Intel Xeon Gold 6338 (Ice Lake).
[**] Intel Core i7-13700K CPU (Raptor Lake).

We also gain greater flexibility in prime selection in the 2D SQIsign setting, since we can use primes of the form $p = c \cdot 2^f - 1$, where c is a small odd integer. Such primes provide abundant two-torsion and enable faster arithmetic, which in turn can further accelerate verification. We also note that recent optimised implementations show that fast 2D verification is competitive in practice, with timings as low as 5.1 million Raptor Lake clock cycles. Moreover, we find that the 2D signing procedure is naturally more amenable to parallelisation [17]. The main computational bottleneck in these 2D schemes, however, remains the evaluation of a 2D isogeny of de-

gree approximately $(2^\lambda, 2^\lambda)$, where λ denotes the security parameter; see [5, Algorithm 8].

3.2 The SQIsign identification scheme

The original SQIsign protocol [11] combines elliptic curves, quaternion algebras, and the Deuring correspondence to build an identification scheme with exponential challenge space, which the Fiat–Shamir transform [18] then turns into a signature scheme with extremely compact signatures and public keys. At its core lies a Σ-protocol that proves knowledge of the endomorphism ring of the public-key curve. Since verification is the relevant SQIsign primitive for this work, we give an informal overview of the protocol, according to its Round 2 NIST specification [1].

Referring to Figure 2, let E_0 be a fixed starting elliptic curve with public endomorphism ring $O_0 = \mathrm{End}(E_0)$. Alice's secret key consists of a randomly chosen ideal I_φ, which the Deuring correspondence translates into an isogeny $\varphi : E_0 \to E_A$, where the codomain E_A serves as her public key. The function IdealToIsogeny, which plays a central role in key generation, computes this isogeny efficiently.

In the Σ-protocol shown in Figure 2, Alice proves knowledge of $\mathrm{End}(E_A)$ as follows. She first generates a random commitment curve E_1 using the same approach as in key generation. Bob, acting as the verifier, then samples a random cyclic 2-isogeny $\varphi_{\mathrm{chall}} : E_1 \to E_2$ and sends a description of φ_{chall} to Alice. Since only Alice knows $\mathrm{End}(E_1)$, only she can efficiently compute $\mathrm{End}(E_2)$. At that point, she has the crucial ingredients, namely, $\mathrm{End}(E_A)$ and $\mathrm{End}(E_2)$, needed to compute a cyclic isogeny $\varphi_{\mathrm{resp}} : E_A \to E_2$.

In the round-two version of SQIsign [1], the central routine IdealToIsogeny relies on Kani's lemma and 2-dimensional abelian varieties, closely following the approach of [5]. More recent work has further accelerated this step through the Qlapoti procedure [8], which also uses 2-dimensional abelian varieties.

The response sent to the verifier is represented as a 2-dimensional isogeny that embeds the 1-dimensional isogeny from the public key to the commitment. Concretely, the response consists of a single 2D isogeny of length approximately $2^{\frac{\lambda}{2}}$ together with a challenge isogeny of length $2^{2\lambda}$. One can compute the 2D isogeny using optimal strategies together with the so-called gluing and splitting isogenies [13].

This structure makes SQIsign implementations inherently asymmetric. Key generation and signing depend both on $\mathbb{F}_p$-arithmetic for isogeny computations and on multi-precision integer arithmetic for quaternion algebra operations, whereas verification uses no quaternion arithmetic at all. Consequently, verification has substantially smaller code coverage.

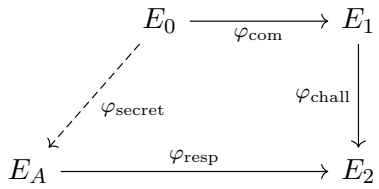


Figure 2: Diagram of the SQIsign protocol.

4 A new elliptic curve

Using standard point-counting software, we find the twist-secure Edwards curve Ed376 defined over the isogeny-friendly field associated with the prime modulus $p = 65.2^{376} - 1$

$$x^2 + y^2 = 1 - 66524x^2y^2$$

There is a generator point corresponding to $x = 2$, which generates the group of prime order q

$$q = 1040000000000000000000000000000000000000000000000303A69B3514879CD109A98F29F0D04F09F855D4F3C6A7037_{16}$$

where the number of points on the curve is $4q$. Since $p = 3 \bmod 4$ the number of points on the twisted curve is also four times a prime.

An Edwards signature based on this curve requires 48 bytes for the public key and 96 bytes for a signature. The implementation basically follows that of the EdDSA method based on the Ed448 curve as specified in the NIST standard document [35]. Observe that since the prime p is less than 384 bits in length, the top bit can be used to encode the sign of the compressed public key, without requiring an additional byte.

If required, there is also the associated twist-secure Montgomery curve

$$y^2 = x^3 + 266098.x^2 + x$$

5 Hybrid Signatures

Just as with KEMs it makes a lot of sense in the near-term to use hybrid signatures. That is to sign everything twice, once with a tried and tested classical scheme, and again with a CRQC-immune post-quantum scheme. Both signatures would need to verify for the overall signature to be accepted. The problem is to decide which methods to pair, and a concern is that given the number of possible combinations TLS could end up having to support multiple proposals. Working out the exact security parity between possible pairings could also be an issue. See [7] for consideration of some possible

combinations. In this recent draft, no less than 18 combinations involving ML-DSA as the post-quantum component are suggested [36].

5.1 Hybrid Signature Requirements in TLS Depend on the Role

One important aspect to take into account in post-quantum authentication in TLS is that there are several TLS signature problems. Typically, a first connection in the Web PKI involves several signatures with different performance and deployment constraints. In the case of a unique or single visit to a website, the client-server interaction may involve not only the certificate chain and the TLS handshake signature, but also additional so-called certificate transparency signatures [45].

We note that there exist two scenarios that are especially relevant. First, some signatures are accompanied by their public keys during the handshake, whereas others are not. On the other hand, for the signature of the root on the intermediate certificate, and for signed certificate timestamps (SCTs), the public key is typically not transmitted during the handshake. In these cases, signature size is the dominant metric. By contrast, the leaf certificate and the TLS transcript signature do involve transmitted public-key material, so for those roles the main design factor to be taken into account is the combined size of the public key and signature.

Second, some signatures are produced offline while others are produced online. Certificate signatures are generated ahead of time and then verified many times by clients. However, the TLS transcript signature must be computed during the handshake protocol and is therefore critical for the overall latency of the protocol. For that reason, the online handshake signature requires a better balance between signing and verification time, whereas the offline certificate signatures can tolerate a slower signing, and even non-constant time signing, if in exchange they offer compact sizes and efficient verification.

Hence, in this paper we suggest that it is better to view TLS authentication as a combination of signatures, where different signatures in the handshake have different constraints on bandwidth, latency and verification cost. Today, the certificate material is a substantial fraction of the data transferred on non-resumed connections, and a naive blunt migration to large post-quantum signatures risks producing certificate chains that are prone to compatibility and performance degradation issues.

5.2 A hybrid authentication design for TLS

Taking into account the observations from the previous subsection, we propose a hybrid authentication strategy rather than a pure post-quantum replacement. Our main idea is to use a fused signature built from SQIsign-3

Table 3: Design approaches for post-quantum TLS authentication.

Approach	**Advantages**	**Size**	**Online signing**	**Main concern**
All-ML-DSA	Uniform replacement across chain and handshake	High $\approx +15\,\mathrm{kB}$	Simple	Certificate and handshake bloat
FN-DSA split	Offline certificates; classical/PQ mix for the handshake	Medium $\approx +7\,\mathrm{kB}$	Delicate	Constant-time floating-point signing
Merkle Tree Certificates	Large-scale WebPKI redesign	Low online bytes	N/A	Significant ecosystem and standards changes
Proposed fused SQIsign-3/Ed376	Compact offline certification material	Low–medium	Avoided	SQIsign verification cost

and an Edwards signature using the companion curve presented in section 4.

We motivate this choice as follows. The 192-bit security level is appropriate for intermediate and root certificates, SQIsign-3 offers remarkably compact post-quantum public keys and signatures, and the Edwards companion curve provides a classical cryptographic hedge over the same base field. By working over the same field, we allow both components to share the underlying field arithmetic. Our resulting fused public key has size

$$97 + 48 = 145 \text{ bytes},$$

and our fused signature has size

$$224 + 96 = 320 \text{ bytes}.$$

These sizes are small enough to make our approach attractive in the parts of TLS where authentication bytes are most expensive.

In our prototype implementation, we deliberately do *not* use SQIsign for the online TLS transcript signature. Instead, as described below, we keep a level-1 hybrid ML-DSA/Ed25519 public key in the leaf certificate, while key exchange uses the de-facto standard X25519MLKEM768 hybrid KEM.

We present this proposal as a proof-of-concept experiment rather than as a fully standardised X.509 implementation. In the current prototype, we represent fused public keys and fused signatures by direct concatenation. A full standardisation effort is still needed to specify the exact semantics of the certificate encoding and the remaining formal design aspects.

Table 3 compares our proposal with other potential TLS design solutions.

5.3 Security semantics of the fused construction

In this work, we intend the fused approach to achieve a *same-message, all-components-must-verify* composition of two signature schemes. We make this semantics explicit, while deliberately keeping the discussion high-level. We focus only on the *signature-composition semantics* of the proposed hybrid design, and we do not attempt to model revocation, CA compromise, side-channel leakage, or the broader standardisation questions related to X.509 encoding. These issues are clearly important, but they are orthogonal to our central claim: under a conjunctive same-message verification rule, the proposed fused construction has the expected hybrid-authentication semantics.

The notion of existential unforgeability under chosen-message attack (EUF-CMA) goes back to the seminal work of Goldwasser, Micali, and Rivest [20], and later it was formalized by Bellare and Rogaway [6].

Intuitively, EUF-CMA security means that given the public key and oracle access to signatures on adaptively chosen messages, an efficient adversary still cannot output a valid signature on a fresh message outside its signing queries.

5.3.1 Security framework

Let

$$\Sigma_1 = (\mathsf{Kg}_1, \mathsf{Sign}_1, \mathsf{Vrfy}_1) \qquad \text{and} \qquad \Sigma_2 = (\mathsf{Kg}_2, \mathsf{Sign}_2, \mathsf{Vrfy}_2)$$

be two digital signature schemes. In our setting, Σ_1 will be a post-quantum scheme (for example SQIsign or ML-DSA, depending on the TLS role) and Σ_2 will be a classical scheme (for example Ed376 or Ed25519).

Definition 1 (Fused signature scheme)**.** *The fused signature scheme*

$$\Sigma_\otimes = (\mathsf{Kg}_\otimes, \mathsf{Sign}_\otimes, \mathsf{Vrfy}_\otimes)$$

is defined as follows.

1. ***Key generation.*** *Execute,*

$$(pk_1, sk_1) \leftarrow \mathsf{Kg}_1, \qquad (pk_2, sk_2) \leftarrow \mathsf{Kg}_2,$$

   and output

$$pk_\otimes = (pk_1, pk_2), \qquad sk_\otimes = (sk_1, sk_2).$$

2. ***Signing.*** *Let* $\mathsf{enc}(\rho, m)$ *be a canonical encoding of the role tag* ρ *and message* m*. On input* $(sk_\otimes, \rho, m)$*, compute*

$$\sigma_1 \leftarrow \mathsf{Sign}_1(sk_1, \mathsf{enc}(\rho, m)), \qquad \sigma_2 \leftarrow \mathsf{Sign}_2(sk_2, \mathsf{enc}(\rho, m)),$$

   and output

$$\sigma_\otimes = (\sigma_1, \sigma_2).$$

3. ***Verification.*** *On input* $(pk_\otimes, \rho, m, \sigma_\otimes)$*, where* $\sigma_\otimes = (\sigma_1, \sigma_2)$*, accept if and only if*

$$\mathsf{Vrfy}_1(pk_1, \mathsf{enc}(\rho, m), \sigma_1) = 1 \quad \textit{and} \quad \mathsf{Vrfy}_2(pk_2, \mathsf{enc}(\rho, m), \sigma_2) = 1.$$

The role tag ρ provides domain separation. In our TLS setting, it distinguishes at least the following cases:

$$\rho \in \{\mathsf{root\text{-}cert}, \mathsf{intermediate\text{-}cert}, \mathsf{leaf\text{-}cert}, \mathsf{tls\text{-}transcript}\}.$$

For certificates, m is the exact `TBSCertificate` byte string. For the TLS 1.3 handshake, m is the standard TLS transcript input signed by the server. Thus both components sign *exactly the same canonical byte string*, under an explicit role tag.

Proposition 1 (Unforgeability of the fused construction)**.** *Suppose* Σ_1 *and* Σ_2 *are EUF-CMA secure. Then* $\Sigma_\otimes$ *is EUF-CMA secure. More precisely, for every probabilistic polynomial-time adversary* $\mathcal{A}$ *against* $\Sigma_\otimes$*, there exist probabilistic polynomial-time adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathrm{Adv}^{\text{euf-cma}}_{\Sigma_\otimes}(\mathcal{A}) \le \mathrm{Adv}^{\text{euf-cma}}_{\Sigma_1}(\mathcal{B}_1)$$

and

$$\mathrm{Adv}^{\text{euf-cma}}_{\Sigma_\otimes}(\mathcal{A}) \le \mathrm{Adv}^{\text{euf-cma}}_{\Sigma_2}(\mathcal{B}_2).$$

Furthermore, a successful forgery against the fused construction yields a successful forgery against each component scheme on the same fresh encoded message.

Proof sketch. We describe the reduction to Σ_1; the reduction to Σ_2 is analogous. Adversary $\mathcal{B}_1$ receives the public key pk_1 and oracle access to $\mathsf{Sign}_1(sk_1, \cdot)$. It generates $(pk_2, sk_2) \leftarrow \mathsf{Kg}_2$ on its own and gives (pk_1, pk_2) to $\mathcal{A}$ as the fused public key.

Whenever $\mathcal{A}$ requests a fused signature on (ρ, m), $\mathcal{B}_1$ queries its signing oracle on $\mathsf{enc}(\rho, m)$ to obtain σ_1, computes $\sigma_2 \leftarrow \mathsf{Sign}_2(sk_2, \mathsf{enc}(\rho, m))$ itself, and returns (σ_1, σ_2).

Suppose $\mathcal{A}$ outputs a valid forgery

$$(\rho^\star, m^\star, (\sigma_1^\star, \sigma_2^\star))$$

for a fresh message such that $(\rho^\star, m^\star)$ was never submitted to the fused signing oracle and

$$\mathsf{Vrfy}_\otimes\big((pk_1, pk_2), \rho^\star, m^\star, (\sigma_1^\star, \sigma_2^\star)\big) = 1.$$

Then in particular

$$\mathsf{Vrfy}_1(pk_1, \mathsf{enc}(\rho^\star, m^\star), \sigma_1^\star) = 1,$$

and freshness of $(\rho^\star, m^\star)$ implies that $\sigma_1^\star$ is a valid EUF-CMA forgery for Σ_1. □

Proposition 1 captures the basic security intuition behind the fused construction: because verification requires *both* components to verify on the *same* message, an attacker who breaks only one component does not automatically obtain a forgery against the fused scheme. In this sense, the fused construction inherits the security of each of its components, provided verifiers enforce the conjunctive verification rule of Definition 1.

Remark 1 (What happens if one component fails?)**.** *Suppose one component scheme becomes insecure, while the other remains EUF-CMA secure. Under the all-components-must-verify rule, the fused construction still cannot be forged unless the remaining secure component is also forged. Thus the fused construction provides the usual hybrid "security hedge" sought in certificate applications.*

Role-specific interpretation in our TLS design. The proposed TLS instantiation uses different signature mechanisms for different authentication roles.

- For *root* and *intermediate* certificate signatures, we use the fused construction based on SQIsign-3 and Ed376. A certificate is accepted only if both component signatures verify on the same `TBSCertificate` encoding.
- For the *leaf* certificate and the *online TLS transcript signature*, we use a level-1 hybrid ML-DSA/Ed25519 public key and the corresponding conjunctive verification rule.
- Therefore, every mandatory authentication step in the non resumed handshake is interpreted under an explicit all-components-must-verify semantics.

Proposition 2. *Consider a non resumed TLS 1.3 handshake in our setting, and assume:*

1. *the client performs standard certificate-path validation;*
2. *each mandatory signature check in the chain and transcript uses the conjunctive verification rule described above;*
3. *the relevant private signing keys have not been compromised; and*
4. *the underlying signature schemes used in each role are EUF-CMA secure.*

Then any successful server-impersonation attack on a fresh handshake implies at least one of the following events:

1. *a successful forgery of a mandatory certificate signature in the accepted chain; or*

2. *a successful forgery of the mandatory transcript signature under the accepted leaf public key.*

Proof sketch. Under standard TLS 1.3 authentication semantics, acceptance of the server identity requires both an accepted certificate chain and a valid transcript signature under the authenticated leaf key. Therefore an impersonating adversary must either cause the client to accept a certificate chain that was not legitimately issued, or else produce a valid transcript signature for the authenticated leaf key without access to the corresponding private key. These are exactly the two events stated above. □

6 A proposal

A fairly obvious idea would be to implement a fused hybrid scheme based on a level 3 SQIsign implementation combined with an Edwards signature based on Ed376. Note that this implementation could share highly optimised finite field arithmetic code for the prime $p = 65.2^{376} - 1$. The combination would hit the 192-bit security level suitable for intermediate and root certificates. The combined public key size would be 97+48=145 bytes, and the combined signature size would be 224+96=320 bytes.

6.1 TiigerTLS

TiigerTLS [41] is a small footprint implementation of TLS1.3 [38] which provides a TLS client written in C++ and Rust, and a TLS server written in Rust. It does not support versions of TLS earlier than 1.3. Session resumption is fully supported. It is primarily aimed at Internet of Things applications, and so does not allocate any heap memory. It was designed from the ground up with agility in mind, and hence easily adapts to novel cryptographic primitives via its Security Abstraction Layer (SAL), a feature that proved invaluable for this exercise. The distribution includes some tools for certificate chain construction.

6.2 An implementation

We have implemented this proposal using TiigerTLS. We use the tools provided there [1] to create a typical X.509 certificate chain which uses SQIsign-3/Ed376 fused signatures as described above. The signatures and public keys are concatenated following the simple method used by the Open Quantum Safe (OQS) project [42]. The leaf certificate uses a hybrid

[1] `https://github.com/Crypto-TII/TLS1.3`

level 1 ML-DSA/Ed25519 public key. The test implementation uses the X25519MLKEM768 hybrid KEM.

We use this implementation to measure the actual traffic for a full handshake between client and server, see Table 4. The server-to-client flow is the most relevant protocol exchange, since it involves the transmission of the certificate chain, authentication material, and the server contribution to key exchange. In our prototype, this material requires only 6643 bytes, which keeps the handshake comfortably below the rough 10kB region at which larger certificate chains begin to become an issue. In a modern browser the compute time is imperceptible to the client (although we acknowledge that SQIsign verification is still slow compared with competing schemes [46] and renders the Ed376 verification time negligible in comparison).

Table 4: TLS 1.3 handshake bandwidth

Phase	**Bytes transmitted**
Client to Server Hello	1450
Server to Client	6643
Client to Server Finish	64

In Table 5 we compare four configurations: classical TLS, a pure ML-DSA authentication, an FN-DSA-based split design, and the fused SQIsign-based design proposed here. One can see that a selective fused hybrid design can reduce the bandwidth cost of post-quantum TLS authentication while preserving the typical structure of TLS 1.3.

7 Conclusion

TLS supports multiple configurations, and many of these would affect our findings. For example, optional client side PKI authentication would greatly increase the client to server traffic. And other choices of PQ signature algorithm for the leaf certificate (like FN-DSA) would further reduce server-to-client traffic. Another contributing factor could be the adoption of certificate compression [19].

Nevertheless, we conclude that an open mind should be kept regarding the future of TLS in the context of the possible emergence of a CRQC. TLS will need to change, but premature and ultimately unnecessary changes, which impact agility, could do more harm than good.

Appendix A: Example Certificate Chain

A leaf plus intermediate certificate chain based on SQIsign-3/Ed376 signatures

Table 5: Approximate size comparison across candidate TLS authentication configurations.

Configuration	**Cert-chain bytes**	**Server→client bytes**
Classical NIST-ECC	≈ 354	$\approx 1,538$
Pure ML-DSA	9,882	13,422
FN-DSA comb	$\approx 5,250$	$\approx 7,036$
SQIsign–Ed376	$\approx 2,129$	**6,643**

Notes.

(a) Classical baseline using the NIST-ECC sizes from Table 1: leaf certificate $65 + 96$, intermediate certificate $97 + 96$, transcript signature 64, and hybrid KEM share 1120.

(b) As analyzed in subsection 2.3: leaf certificate $1312 + 3309$, intermediate certificate $1952 + 3309$, transcript signature 2420, and hybrid KEM share 1120.

(c) Using FN-DSA Level 1 for the leaf and Level 5 for the intermediate/root as presented in Table 1.

(d) Server-to-client bytes are reported in Table 4. The cert-chain value is a first order derivation assuming that we go for an ML-DSA/Ed25519 leaf certificate and SQIsign-3/Ed376 for the intermediate/root.


```
-----BEGIN CERTIFICATE-----
MIIHnTCCBkmgAwIBAgIQNUtikdwKFipyZhN8Hg9YwjAJBgUrzg8MAQUAME4xCzAJ
BgNVBAYMAkFFMRswGQYDVQQKDBJUaWkgVHJ1c3QgU2VydmljZXMxIjAgBgNVBAMM
GVRpaWdlclRMUyBpbnRlcm11ZGlhdGUgQ0EwHhcNMjYwMzIyMDAwMDAwWhcNMjYx
MDA2MDAwMDAwWjBFMQswCQYDVQQGDAJBRTEbMBkGA1UECgwSVG1pIFRydXN0IFNl
cnZpY2VzMRkwFwYDVQQDDBBUaWlnZXJUTFMgc2VydmVyMIIFUjAHBgUrzg8MBgOC
BUUAAAAAIHLMSqvr/ImZZ46F3+5/YWO+JkJRIzixQDyKPivfgc2z+mQciJue/+nn
YcipveWdTopJSxb1Q2SpLtbQR/+nPOSGZKJgrYek7oFP32GjZZJQMaMIOZk27cr0
TG5To9cuf6neLuXQHJ9oUHLEmsI/I9ptJJm7iDkkAUzdalB2Br4FYt6afcZR4Tmg
z12OmTxrfhD+k2oOOS3bIh8FUQvI3964XkOBIhMUymsgTDnN8xSjiEC7yMnQu3Ao
3u+oVn3rHXBu6u+kNOD3OKUVGsL1QVZBvRgYfAwN/PD5ZQjZc3cnROGhCS+2KSK7
ymOOwFKdTrPNKRP5FKZNjftZTJ8Z7C5SuzzvEKdv6hjUqE2rmdRUsVbAcVxnsW2B
Qh+FwbiyyA3TOOEPxJcC+LVSXsecroAX6U8Uz4QwFbVJtXQ3POCyU233FWyejl/h
SOWXGpIwl/NOVFiociKlvz6AkZmTfayp4D9YA5yUSrpQyOi/kSANF/SV+KwBYJbY
r6vwvJQDTkd5NTYh1BqaTiMU53cQcpiYOHU2SY9v1UNHgPRqX6oVVJheZWdPx/Jo
vggSc8baLMpTCEeTLo267j/U26pK4pGYbhH5zbsZZCPzvd7oHUNSsTFo6Rh1N7yA
eAEVMBoOv7jSjSBsiNKuQnV/UhEtQTX1htPBCPk9Qc0PEHS8ioWvnd+7k1xlxhe1
B9SgWgqDTa9mvg+gC54PGgG3hDv+eq6DpZklEbKci37zFB4v9bl228gdAq4lRlw5
k8UPwdKMuAGl1YuN1LXmhdggAWMx1f7BQlBAzXfiTxnq5klmhtOPjQDVpzxLxMKM
Z692DXBCJgaZrraPKG4cVpGsfJ4ohbIuGNOvWaiyOlL6RObYwX+8KXjCPBhBX/vk
qYnUzLgPFktfXJlHcvMHf8rcQLjpB8l+N3lecGsCiUsOIS4ZHuy4iZPF16d6/+wL
jLDd86l61hljo3cLvfEyi4LeaXxh9xjHtq48WLE1ZoUJdb0YP6CnWlvUepOhFLp9
/V3NnKTp1R9liETsdjOxfUx0gWegYGLvtXe8kBPXh9OWMxCScZwrrwoAL6DNIkpw
O7ihv+e+UHiqO+4rIBd8YepGJ1rVBMO+MZn9ZyV+vwfw1UzAMlKb3shehLzVd6BM
z7huXI6qRrqQfzpYt2f6G6EGJRudrEaVT/uxCeIIdK6W7JkMbRjMjGptDEJcQ2XC
ucJv6ho/eVCq3AujAIPFzxwairxoGOKU3D6e7Z27jjxpcoFAwC9POju5E7tpUbv/
Q6rcDelKfl3BAm37O2+JZCtyUYU9CEIZ3CdHSCT39ChbN/2x6kdcvMhlG1+GwVbo
ApApqy5RddHFL6q6j20RuWEraI6Jb+UpEIEBe/XN3Hh4S+7MT8mRrU+DT8AOhOi9
2WYmqGjZeWYHxdSKVyjx1NykqK9kK1ZXUg6HykJNxmlVyz4j9knaGyu1Lf9w4VZE
Bh1jtmwIoSkVxgoCe9WSeYWCtHwLyORBMQOt6+QyP/KlYiYJxqqBV4rnHPARyubN
6K5H1/zX2kLNvX7Sr4cTy/QWpn19USkSTY9sARoaNgpVmp9SRpaiGuFtsoR24oJO
6tMo3qye2Sl57bhNlRZb2bW0i6ebqvUov/JmhEmzF5ev9n5VVDcWIjHUvbbd3lF9
sNLBDlxngPa0uQxNfCpqDZ9Tkx838eUVbG1tJx+MO7dOjZb6IdvqadmDQk1OjqD5
OfQfRpZn9ctSshGB6MaOBJNeix/foWNjgei6xqwNb8rYcLQp14j21ZvZLcA5ZNUT
KneLOVme8KMYMBYwFAYDVRORBA0wC4IJbG9jYWxob3NOMAkGBSvODwwBBQADggFB
AIJ7MB+O8MCruh8skhq7BDvGvGi/KoH7alTlNthuZf+fgiB7aBD2/WvGLxK3zlir
IbBfyyfE6vT1qQZFRTqihPUxyKt66H3uOU1mo3QUoTpfZDPCROHzbyqQP9vRBJpm
C9q182ljRAR5fAP8qOF3sRaO4XCc6KTKdxo9BpTKfBoNzgOiKx9BZ5Cy1JmM6+1p
IY/DSG/Pn3ciL52tzKMLemjCGf2zkgFK1AIyvxag5Dc3R4Hx+miAaCgJJoV3mCmi
IAADiuPKD7yAl1vfTlG/NB4/T+pH16rPfK6gAbe5Mspg2ODCkmo7qBXmLQNZ5J/J
i9z+5wOKDwNjIuOugWyoHUZVGbeXPdCiB272XOoAuOOOodzbynxRxKyuace/RsW5
oidZpLvvAPG5bfAjVIUOOzrdh4MXWs4bpvlpUtVuAAYC
```

```
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC6DCCAZSgAwIBAgIQBnYkLfv9B+5Qm9ecDRVPdjAJBgUrzg8MAQUAMEYxCzAJ
BgNVBAYMAkFFMRswGQYDVQQKDBJUaWkgVHJ1c3QgU2VydmljZXMxGjAYBgNVBAMM
EVRpaWdlclRMUyByb290IENBMB4XDTI2MDMyMjAwMDAwMFoXDTI2MTAwNjAwMDAw
MFowTjELMAkGA1UEBgwCQUUxGzAZBgNVBAoMElRpaSBUcnVzdCBTZXJ2aWNlczEi
MCAGA1UEAwwZVGlpZ2VyVExTIGludGVybWVkaWF0ZSBDQTCBojAHBgUrzg8MAQOB
lgAAAAAwtuoH6bk6Mc9dIIJdzoqo+nBqOy90aremsn8QokCpwBOKpkVGgZ/V1eOO
duID+Vsj73y6UfPv5n6RQAH4sSRpvGFi8lGM6RnNDA32wPltaQeJPdTP7NpitdRQ
8lw3lMAtd2GNIS2qMvpPbt0bTK9yszMfbo6aoC9ZSsiuyCqfYgFKMt+HVkgVgZBK
E9v5sygyAqMTMBEwDwYDVROTAQH/BAUwAwEB/zAJBgUrzg8MAQUAA4IBQQDYjURm
CHJnt2+Cwiix5jJB3yUlJWV4GpHsMVLqd4OEe/dKzA+Mk/HE8lFroWN2SpdntfBY
uPQr7bwPAfw62xTHwyT8Ps+zaq0L7cmcq43g7tiUlw16uQsevyjn2s3/pgStzWkN
8IkvMS/JDWCkB1WxDKz/ZRs9byKzs8YvS4JPqFvd5vtKcZFFpdO2iLBUyyAgNR3B
LXg/mt0q+shD5QxC+8yr274YjgMZiZGCmHH5jfbH42voSOsEvrfQz7aslhMABdnO
JfXNVxeglPvpiE9ZB7CS2uJsCYMLiQP3PMonap2chOHjjCyuWKp4o6F3cSXcjlQD
E2SI7Dpiux/yr8ARXtE+z0rp2+ZZiYItAP2Vhk97YygaeXI5QYezPxU+QzrWiKDf
IgEBUHO5DPusIfpPfCaAKmGf5I5Ej6AogwANAg==
-----END CERTIFICATE-----
```


References

[1] Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign. Technical report, National Institute of Standards and Technology, 2025.

[2] Olivier Adjonyo, Arwa Alblooshi, Gilbert Dione Isaac Canales-Martínez, Jorge Chávez-Saab, and Francisco Rodríguez-Henríquez. PQ-SORT: Post-quantum signatures on-ramp tests, 2026. `https://pqsort.tii.ae/`.

[3] Jean-Claude Bajard and Sylvain Duquesne. Montgomery-friendly primes and applications to cryptography. *Journal of Cryptographic Engineering*, 11:399 – 415, 2021.

[4] A. Basso, G. Borin, W. Castryck, M. Corte-Real Santos, R. Invernizzi, A. Leroux, L. Maino, F. Vercauteren, and B. Wesolowski. PRISM with a pinch of salt: Simple, efficient and strongly unforgeable signatures from isogenies. Cryptology ePrint Archive, Paper 2026/443, 2026.

[5] Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West - the fast, the small, and the safer. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology - ASIACRYPT 2024, Proceedings, Part III*, Lecture Notes in Computer Science, pages 339–370. Springer, 2024.

[6] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[7] Nina Bindel and Britta Hale. A note on hybrid signature schemes. Cryptology ePrint Archive, Paper 2023/423, 2023.

[8] Giacomo Borin, Maria Corte-Real Santos, Jonathan Komada Eriksen, Riccardo Invernizzi, Marzio Mula, Sina Schaeffler, and Frederik Vercauteren. sfQlapoti: Simple and efficient translation of quaternion ideals to isogenies. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology - ASIACRYPT 2025, Proceedings, Part IV*, Lecture Notes in Computer Science, pages 174–205. Springer, 2025.

[9] J. Bos, C. Costello, P. Longa, and M. Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. Cryptology ePrint Archive, Paper 2014/130, 2014. `https://eprint.iacr.org/2014/130`.

[10] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, 2023.

[11] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign: Algorithm specifications and supporting documentation, 2023. National Institute of Standards and Technology.

[12] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQIsignHD: New dimensions in cryptography. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024, Proceedings, Part I*, Lecture Notes in Computer Science, pages 3–32. Springer, 2024.

[13] Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An algorithmic approach to (2, 2)-isogenies in the theta model and applications to isogeny-based cryptography. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology - ASIACRYPT 2024, Proceedings, Part III*, Lecture Notes in Computer Science, pages 304–338. Springer, 2024.

[14] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2020.

[15] Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. New algorithms for the Deuring correspondence - towards practical and secure sqisign signatures. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 659–690. Springer, 2023.

[16] Max Duparc and Tako Boris Fouotsa. SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. Cryptology ePrint Archive, Paper 2024/773, 2024. `https://eprint.iacr.org/2024/773`.

[17] Luca De Feo, Li-Jie Jian, Ting-Yuan Wang, and Bo-Yin Yang. SQISign on ARM. Cryptology ePrint Archive, Paper 2026/394, 2026.

[18] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[19] Alessandro Ghedini and Victor Vasiliev. TLS Certificate Compression. RFC 8879, 2020.

[20] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[21] Marin Ivezic. Google's Merkle tree (MTC) gambit to quantum-proof HTTPS, 2026. `https://postquantum.com/security-pqc/googles-merkle-tree-mtc-https/`.

[22] D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *PQCrypto 2011*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34, 2011.

[23] Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik*, 485:93–122, 1997.

[24] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Car-

mit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471. Springer, 2023.

[25] Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andrew Rusch. PKCS #1: RSA cryptography specifications version 2.2. RFC 8017, 2016.

[26] Kohei Nakagawa and Hiroshi Onuki. QFESTA: Efficient algorithms and parameters for festa using quaternion algebras. *Cryptology ePrint Archive*, 2023.

[27] Kohei Nakagawa, Hiroshi Onuki, Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon, and Frederik Vercauteren. SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology - ASIACRYPT 2024, Proceedings, Part III*, Lecture Notes in Computer Science, pages 272–303. Springer, 2024.

[28] National Institute of Standards and Technology. Recommendation for key management: Part 1 – general. Technical Report SP 800-57 Part 1 Rev. 5, NIST, 2020.

[29] National Institute of Standards and Technology. Digital signature standard (DSS). Technical Report FIPS 186-5, NIST, 2023.

[30] National Institute of Standards and Technology. Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters. Technical Report SP 800-186, NIST, 2023.

[31] National Institute of Standards and Technology (NIST). Falcon-based digital signature standard (FIPS 206, in development). NIST Post-Quantum Cryptography Standardization Process, 2025. FN-DSA / Falcon standard under development.

[32] National Institute of Standards and Technology (NIST) and David Cooper. Stateless hash-based digital signature standard, 2024.

[33] National Institute of Standards and Technology (NIST), Thinh Dang, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, and Angela Robinson. Module-lattice-based digital signature standard, 2024.

[34] National Institute of Standards and Technology (NIST). Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process, 2022.

[35] National Institute of Standards and Technology. Digital signature standard. Technical Report Federal Information Processing Standards Publications (FIPS) 186-05, February 03, 2023, U.S. Department of Commerce, Washington, D.C., 2023.

[36] Mike Ounsworth, John Gray, Massimiliano Pala, Jan Klaußner, and Scott Fluhrer. Composite Module-Lattice-Based Digital Signature Algorithm (ML-DSA) for use in X.509 Public Key Infrastructure, 2026.

[37] Aurel Page and Damien Robert. Introducing Clapoti (s): Evaluating the isogeny class group action in polynomial time. 2023.

[38] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.

[39] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. RFC 9849, 2026.

[40] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503. Springer, 2023.

[41] Michael Scott. On TLS for the internet of things, in a post quantum world. Cryptology ePrint Archive, Paper 2023/095, 2023.

[42] D. Stebila and M. Mosca. Post-quantum key exchange for the internet and the open quantum safe project. In *Selected Areas in Cryptography (SAC) 2016*, volume 10532 of *LNCS*, pages 1—24, 2017.

[43] Sean Turner, David Brown, Kelvin Yiu, and Russ Housley. Elliptic curve cryptography subject public key information. RFC 5480, 2009.

[44] L. Valenta, C. Patton, V. Goncalves, and B. Westerban. Keeping the internet fast and secure: introducing Merkle tree certificates, 2025. `https://blog.cloudflare.com/bootstrap-mtc/`.

[45] B. Westerban. State of the post-quantum internet in 2025, 2025. `https://blog.cloudflare.com/pq-2025/`.

[46] Thom Wiggers. Post-quantum signatures zoo, 2026. `https://github.com/pqshield/nist-sigs-zoo`.