Optimized One-Dimensional SQIsign Verification on Intel and Cortex-M4

Marius A. Aardal[1], Gora Adj[2], Arwa Alblooshi[2], Diego F. Aranha[1], Isaac A. Canales-Martínez[2], Jorge Chávez-Saab[2], Décio Luiz Gazzoni Filho[3,4], Krijn Reijnders[5] and Francisco Rodríguez-Henríquez[2]

[1] Aarhus University, Aarhus, Denmark
{maardal,dfaranha}@cs.au.dk
[2] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
{gora.adj,arwa.alblooshi,isaac.canales,jorge.saab,francisco.rodriguez}@tii.ae
[3] Instituto de Computação, Universidade Estadual de Campinas (UNICAMP), Campinas, Brazil
decio.gazzoni@ic.unicamp.br
[4] Department of Electrical Engineering, State University of Londrina, Londrina, Brazil
[5] Radboud University, Nijmegen, Netherlands
krijn@cs.ru.nl

Abstract.
SQIsign is a well-known post-quantum signature scheme due to its small combined signature and public-key size. However, SQIsign suffers from notably long signing times, and verification times are not short either. To improve this, recent research has explored both one-dimensional and two-dimensional variants of SQIsign, each with distinct characteristics. In particular, SQIsign2D's efficient signing and verification times have made it a focal point of recent research. However, the absence of an optimized one-dimensional verification implementation hampers a thorough comparison between these different variants. This work bridges this gap in the literature: we provide a state-of-the-art implementation of one-dimensional SQIsign verification, including novel optimizations. We report a record-breaking one-dimensional SQIsign verification time of 8.55 Mcycles on a Raptor Lake Intel processor, closely matching SQIsign2D on the same processor. For uncompressed signatures, the signature size doubles and we verify in only 5.6 Mcycles. Taking advantage of the inherent parallelism available in isogeny computations, we present 5-core variants that can go as low as 1.3 Mcycles. Furthermore, we present the first implementation that supports both 32-bit and 64-bit processors. It includes optimized assembly code for the Cortex-M4 and has been integrated with the pqm4 project. Our results motivate further research into one-dimensional SQIsign, as it boasts unique features among isogeny-based schemes.

Keywords: post-quantum cryptography, isogeny, SQIsign, verification, ARM

1 Introduction

In June 2023, NIST initiated Round 1 of the Call for Additional Digital Signature Schemes [52], accepting 40 proposals based on diverse cryptographic problems. Notably, Short Quaternion and Isogeny Signature (SQIsign) is the sole isogeny-based scheme among these candidates. SQIsign offers the smallest combined signature and public-key size of all post-quantum signature candidates. However, SQIsign's signing time is not only considerably slower than lattice-based schemes, but all current implementations are not constant-time [9], while verification remains moderately efficient. For this reason, it has been argued that SQIsign's potential lies in applications requiring long-term signatures, such as certificates, software verification, and resource-constrained devices performing only verification tasks.

Introduced at Asiacrypt 2020, SQIsign [21] has seen intense development since then. While an improved version preserving the original framework was presented in [22], recent research has explored more substantial modifications to the underlying scheme.

Building upon SQIsign, Corte-Real Santos et al. [11] proposed ApresSQI, a variant specifically designed to prioritize faster verification at the cost of having the signer perform extension-field arithmetic. They estimate that ApresSQI's verification can be up to 4× faster than SQIsign's, with only a factor two slowdown to signing. However, ApresSQI lacks an optimized low-level implementation to fully corroborate these claims.

Higher-dimensional techniques, introduced to cryptography in the break of SIDH/SIKE [8, 38, 46], led to the development of SQIsignHD [17], which greatly improves signing speed compared to SQIsign, reporting an astonishing 40× acceleration factor. Key generation also benefits from similar speedups. However, SQIsignHD verification requires the evaluation of a 4- or 8-dimensional isogeny, which makes it significantly more expensive, raising serious concerns about its practicality. Recently, the algorithmic developments introduced in [40, 43] were exploited to create a two-dimensional (2D) variant of SQIsign, independently proposed in [3, 25, 41]. These variants leverage 2D isogeny computations for both signing and verification. Compared to the original and one-dimensional (1D) SQIsign, the performance improvements reported in [3] are nothing less than remarkable: signing achieves a 15× acceleration, and verification enjoys a 4× speedup. Moreover, the security proofs of these 2D variants demonstrate increased rigor, essentially eliminating the heuristic assumptions required in 1D SQIsign.

Another interesting SQIsign variation was proposed by Onuki and Nakagawa in [42], which employs 2D techniques in the core ideal-to-isogeny algorithm to accelerate the costly isogeny computations in the signature procedure of the original SQIsign, while retaining a 1D verification. Like the other 2D variants, this also offers greater flexibility in prime selection, enabling the use of primes of the form $p = c \cdot 2^f - 1$, with c a small odd integer. This enables more efficient verification due to their higher two-torsion and fast arithmetic, as shown in ApresSQI. In principle, their signing algorithm allows for faster verification for 1D SQIsign-like schemes and becomes quite amenable for parallelization, which we explore in this paper.

The primary computational bottleneck in 2D isogeny-based schemes is the computation of a 2D isogeny of degree approximately $(2^\lambda, 2^\lambda)$, with λ the security parameter (see [3, Algorithm 8] for more technical details). Conversely, 1D verification involves the computation of a simpler but longer 2^e-isogeny with $e \approx 7.5\lambda$, decomposed into $n = \lceil \frac{e}{f} \rceil$ blocks of 2^f-isogenies, with f as defined above. In this work, we carefully study how to make 1D verification as efficient as possible, which includes exploring and exploiting its inherent parallelizability.

The rapidly evolving landscape of SQIsign has hindered progress on critical research areas, such as the development of efficient implementations that keep up with the most recent developments and exploring SQIsign's feasibility on resource-constrained platforms like the ARM Cortex-M4. Given SQIsign's primary focus on efficient verification for potentially resource-constrained devices, evaluating its practical deployment readiness is of paramount relevance.

1.1 Our Contributions

This work aims to evaluate the competitiveness of 1D SQIsign verification against the performance gains reported for 2D variants in [3], and to provide a state-of-the-art library for 1D verification with extended compatibility. Our main contributions are as follows:

1. Building upon the *smart sampling* technique introduced in ApresSQI [11], we introduce several new optimizations for both compressed and uncompressed SQIsign. These improvements include a more efficient public-key representation and a streamlined hash to a kernel point. The slower compressed variant is found to be on par with the 2D verification of [3], while the larger-sized uncompressed variant is about 50% faster than the compressed one at the cost of an additional 163 B in signature size for NIST Level I.
2. Exploiting the inherent parallelizability of 1D verification, we present variants of compressed and uncompressed signatures that allow for a 5-core verification. We achieve the first sub-millisecond verification latency (1.33 million Raptor Lake cycles), albeit at the cost of a modest increase in signature size.
3. We present the first optimized 1D SQIsign verification library compatible with both 32-bit and 64-bit architectures. The library incorporates several state-of-the-art types of arithmetic offered by third parties, including a new portable C option, and optimized assembly code for the Cortex-M4. It is integrated with the pqm4 project [33], yielding the first ever SQIsign benchmarks on this platform. Although our speed records are achieved on new parameters, our library also implements the current NIST parameters to remain compatible with its test vectors.

Table 1 summarizes the performance of our SQIsign verification versus previous works.

Table 1: Comparison of execution time of SQIsign schemes at NIST security level 1, in millions of clock cycles. All measurements are on the same platform, an Intel Core i7-13700K (Raptor Lake) server with TurboBoost and hyperthreading disabled. The *ref* columns refer to builds using Fiat-Crypto arithmetic, while *opt* refers to the custom C arithmetic with Intel intrinsics in the case of 2D-West, and assembly code targeting the Broadwell microarchitecture in all other cases.[1]

	Size (bytes)		Verification (ref)		Verification (opt)	
	Public key	Signature	Mcc	Accel.	Mcc	Accel.
SQIsign [9]	64	177	71.42	1.00	34.28	1.00
SQIsign 2D-West [3]	66	148	19.11	3.73	8.43	4.06
Our compressed	64	157	19.03	3.75	8.55	4.00
Our uncompressed	64	320	12.63	5.65	5.60	6.12
Our compressed parallel	64	285	4.72	15.13	2.03	16.88
Our uncompressed parallel	64	448	2.90	24.62	1.33	25.77

Organization. The paper is structured as follows. Section 2 provides a concise overview of elliptic curves, isogenies, and the family of SQIsign variants. As motivation for this work, in Section 3 we provide cost models for 1D and 2D verification that sustain the ongoing relevance of 1D verification. Section 4 details our optimized 1D SQIsign verification techniques. Section 5 discusses the features of our software library, and Section 6 presents a comprehensive performance evaluation together with concluding remarks.

2 Preliminaries

SQIsign leverages both the realm of elliptic curves, as well as the realm of quaternion algebras. We will give a brief overview of both before describing SQIsign itself.

[1]The 2D-West implementation does not incorporate assembly arithmetic targeting the Broadwell microarchitecture, but, as discussed in Section 6, its optimized arithmetic is found to be roughly comparable.

Notation. Throughout this work, we let $p > 3$ denote a prime and $\mathbb{F}_{p^k}$ the finite field with p^k elements, with $k \in \mathbb{Z}_{>0}$. The algebraic closure of $\mathbb{F}_{p^k}$ is $\overline{\mathbb{F}}_{p^k} = \bigcup_{\ell \geq 1} \mathbb{F}_{p^{\ell k}}$.

2.1 Elliptic Curves, Isogenies, and the Deuring Correspondence

We provide here a concise overview of the essential mathematical concepts and definitions underlying the SQIsign protocol. For an in-depth treatment, we refer to [5, 19].

2.1.1 Elliptic Curves

A Montgomery curve over $\mathbb{F}_{p^k}$ is an elliptic curve given by the affine equation [15, 39],

$$E : By^2 = x^3 + Ax^2 + x,$$

for some constants $A,B \in \mathbb{F}_{p^k}$ such that $B(A^2 - 4) \neq 0$. In this paper, we only work with elliptic curves of this form. For any extension $\mathbb{F}_{p^{\ell k}}$, the set of points $(x,y) \in \mathbb{F}_{p^{\ell k}} \times \mathbb{F}_{p^{\ell k}}$ satisfying this equation, together with the point at infinity $\mathcal{O}_E$, form a finite abelian group $E(\mathbb{F}_{p^{\ell k}})$. The group $E(\overline{\mathbb{F}}_{p^k})$ is the collection of all these points. We use additive notation $P+Q$ for the group law. Scalar multiplication by $n \in \mathbb{Z}_{>0}$ is denoted $[n]P = P+P+\cdots+P$. The n-torsion subgroup of E is

$$E[n] = \{P \in E(\overline{\mathbb{F}}_{p^k}) \mid [n]P = \mathcal{O}_E\}.$$

If $p \nmid n$, then $E[n] \cong \mathbb{Z}/n \times \mathbb{Z}/n$. We say that a point P is above another point Q if $Q = [k]P$ for some $k \in \mathbb{Z}$. For later use in Section 4.1, we note that when $E[2] \subseteq E(\mathbb{F}_{p^k})$, we can factor $x^3 + Ax^2 + x$ as $x(x - \alpha)(x - 1/\alpha)$ for some $\alpha \in \mathbb{F}_{p^k}$.

Montgomery curves enable efficient elliptic curve operations using only the x-coordinate of a point through the application of differential addition and doubling [15, 39]. Since P and $-P$ are the only points on the curve with the same x-coordinate, and they generate the same subgroup, it will often be sufficient for us to represent points solely by their x-coordinate, and rely on x-only arithmetic.

Two curves E and E' are isomorphic over a field $K/\mathbb{F}_{p^k}$ if there exists a linear change of coordinates defined over K that is an isomorphism from $E(\overline{\mathbb{F}}_{p^k})$ to $E'(\overline{\mathbb{F}}_{p^k})$. We say that E and E' are isomorphic if they are isomorphic over $\overline{\mathbb{F}}_{p^k}$. This is the case if and only if they have the same j-invariant. For a Montgomery curve E, the j-invariant is $j(E) = 256(A^2 - 3)^3/(A^2 - 4)$ [15], which is independent of B. Because of this, it is customary to describe a Montgomery curve using only its A coefficient, as E_A. A quadratic twist of E is a curve E^t that is isomorphic to E over $\mathbb{F}_{p^{2k}}$ but not over $\mathbb{F}_{p^k}$.

2.1.2 Isogenies

An isogeny $\varphi : E \to E'$ is a non-constant morphism, thus $\varphi(\mathcal{O}_E) = \mathcal{O}_{E'}$ and $\varphi(P + Q) = \varphi(P) + \varphi(Q)$ for all points $P, Q \in E$. This work only uses *separable* isogenies, which are described up to isomorphism by their (finite) kernel. For such isogenies, the degree $\deg(\varphi) = |\ker(\varphi)|$. Given another isogeny $\psi : E' \to E''$, $\deg(\psi \circ \varphi) = \deg(\psi) \cdot \deg(\varphi)$. Conversely, an isogeny of composite degree $d = \prod_i \ell_i^{e_i}$ can be factored as the composition of e_i isogenies of degree ℓ_i for each i. Furthermore, for any $\varphi : E \to E'$ of degree d, there exists an isogeny $\hat{\varphi} : E' \to E$ such that $\hat{\varphi} \circ \varphi = [d]$, the multiplication-by-$d$ map $P \mapsto [d]P$ on E. The isogeny $\hat{\varphi}$ is the *dual* of φ. By Tate's theorem [53], there exists an isogeny defined over $\mathbb{F}_{p^k}$ between E and E' if and only if both curves have the same cardinality over $\mathbb{F}_{p^k}$, i.e., $\#E(\mathbb{F}_{p^k}) = \#E'(\mathbb{F}_{p^k})$. We say that φ is cyclic if $\ker(\varphi)$ is a cyclic group. If E is defined over $\mathbb{F}_{p^k}$, we say that φ is rational if $\ker(\varphi) \subseteq E(\mathbb{F}_{p^k})$.

A description of E and $\ker(\varphi)$ (i.e. the x-coordinate of the generators) is enough to compute E'; we call this procedure xISOG. This description is also sufficient to compute the image $\varphi(Q)$ of a point Q; we call this procedure xEVAL.

2.1.3 Endomorphism Rings and Supersingular Elliptic Curves

An endomorphism $\varphi : E \to E$ is an isogeny from a curve E to itself or the zero map, i.e, $\varphi(E) = \{\mathcal{O}_E\}$. Two important examples are the scalar multiplication map $[n]$ and, when E is defined over $\mathbb{F}_{p^k}$, the p^k-power Frobenius endomorphism $\pi : (x,y) \mapsto (x^{p^k}, y^{p^k})$. The endomorphisms of E form a ring $\mathrm{End}(E)$ under the addition and composition of endomorphisms. The curves with the largest endomorphism rings are called *supersingular*; their endomorphism rings are isomorphic to maximal orders in the quaternion algebra $\mathcal{B}_{p,\infty}$ over $\mathbb{Q}$ ramified at p and ∞.[2]

From this point on, we only work with supersingular elliptic curves, and usually only up to isomorphism. Over characteristic p, the isomorphism class of any supersingular curve has a representative E that is defined over $\mathbb{F}_{p^2}$, whose p^2-power Frobenius map equals $[-p]$. The group structure of E and its quadratic twist over $\mathbb{F}_{p^{2k}}$ is fully characterized by

$$E(\mathbb{F}_{p^{2k}}) \cong \mathbb{Z}/(p^k-(-1)^k)\times\mathbb{Z}/(p^k-(-1)^k), \qquad E^t(\mathbb{F}_{p^{2k}}) \cong \mathbb{Z}/(p^k+(-1)^k)\times\mathbb{Z}/(p^k+(-1)^k).$$

Observe that if n divides $p^k + 1$ or $p^k - 1$, then the n-torsion is fully defined over $\mathbb{F}_{p^{2k}}$, either on E or its twist.

The security of most isogeny-based cryptography relies on the conjectured hardness of the isogeny problem: Given two supersingular curves E_1, E_2 over $\mathbb{F}_{p^2}$, find an isogeny between them. The best known classical attack [23] runs in time $\tilde{O}(\sqrt{p})$ and the best quantum attack [6] runs in $\tilde{O}(\sqrt[4]{p})$. Another problem of interest is the endomorphism ring problem: Given a supersingular elliptic curve E, compute an efficient representation of $\mathrm{End}(E)$. It was shown in [44, 55] that these are equivalent. Additionally, [44] shows that the endomorphism ring problem is equivalent to the one endomorphism problem, which asks to find just one non-scalar endomorphism of E.

2.1.4 The Deuring Correspondence and KLPT

We mentioned that for a supersingular curve E, its endomorphism ring $\mathrm{End}(E)$ is isomorphic to $\mathcal{O}$, where $\mathcal{O}$ is a maximal order in the quaternion algebra $\mathcal{B}_{p,\infty}$. In fact, the Deuring correspondence [24] tells us that there is an equivalent view of isogenies in the quaternion world. The set of maximal orders in $\mathcal{B}_{p,\infty}$ under isomorphisms is in bijection with the set of supersingular j-invariants over $\mathbb{F}_{p^2}$ (under Galois conjugacy). A separable isogeny $\varphi : E \to E'$ of degree d corresponds to an integral ideal I_φ of norm d, with left order $\mathcal{O}$ and right order $\mathcal{O}' \cong \mathrm{End}(E')$. We then say that I_φ is a connecting ideal between $\mathcal{O}$ and $\mathcal{O}'$. Conversely, every left integral $\mathcal{O}$-ideal corresponds to an isogeny from E.

Remarkably, problems that seem hard in the isogeny world become easy in the quaternion world. The quaternion equivalent of the isogeny problem is the following: Given two maximal orders $\mathcal{O}, \mathcal{O}'$, compute a connecting ideal (of smooth norm). This can be solved in PPT using the (generalized) KLPT algorithm [22, 34]. This algorithm takes as input an ideal I and outputs an equivalent ideal J of a desired smooth norm, with sufficient success probability when the norm requested is about $15\log_2(p)/4$ bits long. The smooth-norm ideal can be translated back to a smooth-degree isogeny, and efficiently computed as the composition of several small-degree isogenies. Thus, the only thing preventing an attacker from solving the isogeny problem is the lack of knowledge of the endomorphism rings.

2.2 SQIsign

The original SQIsign protocol [21] unites the worlds of elliptic curves and quaternion algebras using the Deuring correspondence into a cleverly constructed identification scheme

[2]This is one of many equivalent definitions of supersingular curves. Another is that $E[p^\ell] = \{\mathcal{O}_E\}$ for all $\ell \in \mathbb{Z}_{>0}$.

$$\begin{tikzcd}
E_0 \arrow[r, "\varphi_{\mathrm{com}}"'] \arrow[d, dashed, "\varphi_{\mathrm{secret}}"] & E_1 \arrow[d, "\varphi_{\mathrm{chall}}"'] \\
E_A \arrow[r, "\varphi_{\mathrm{resp}}"] & E_2
\end{tikzcd}$$

Figure 1: Diagram of the SQIsign protocol.

with exponential challenge space, and, using the Fiat-Shamir transform [28], into a signature scheme with remarkably small signatures and public keys. The underlying Σ-protocol proves knowledge of the endomorphism ring of the public key E_A. We give an informal description of the Σ-protocol, based on the specifications submitted to NIST [9], paying special attention to the verification protocol, the main focus of this work.

2.2.1 SQIsign Identification Scheme

We assume a starting elliptic curve E_0, whose endomorphism ring $\mathrm{End}(E_0)$ is public. Alice's secret key is an isogeny $\varphi_{\mathrm{secret}} : E_0 \to E_A$ that maps E_0 to her public key curve E_A. Knowledge of $\varphi_{\mathrm{secret}}$ allows Alice to efficiently compute $\mathrm{End}(E_A)$. In the following Σ-protocol, portrayed in Figure 1, she proves knowledge of this endomorphism ring.

Commitment. Alice generates a random isogeny $\varphi_{\mathrm{com}} : E_0 \to E_1$ and commits to E_1. The isogeny φ_{com} must remain secret, to prevent Bob from recovering $\mathrm{End}(E_A)$.

Challenge. Bob samples a random cyclic isogeny $\varphi_{\mathrm{chall}} : E_1 \to E_2$ and sends a description of φ_{chall} to Alice. As only Alice knows $\mathrm{End}(E_1)$, only she can compute $\mathrm{End}(E_2)$.

Response. Alice then has the right ingredients, namely $\mathrm{End}(E_A)$ and $\mathrm{End}(E_2)$, to compute a cyclic isogeny $\varphi_{\mathrm{resp}} : E_A \to E_2$ using KLPT. She computes it such that $\hat{\varphi}_{\mathrm{chall}} \circ \varphi_{\mathrm{resp}}$ is cyclic and of a specific degree ℓ^k, and sends φ_{resp} to Bob.[3]

Verification. Bob verifies that φ_{resp} is indeed an isogeny $E_A \to E_2$ of degree ℓ^k, and that $\hat{\varphi}_{\mathrm{chall}} \circ \varphi_{\mathrm{resp}}$ is cyclic.

It is shown in [21] that the Σ-protocol is special sound with respect to the one-endomorphism relation

$$R = \{(E_A, w) \mid w \text{ is a non-scalar smooth-degree endomorphism of } E_A\}.$$

This proof requires that $\hat{\varphi}_{\mathrm{chall}} \circ \varphi_{\mathrm{resp}}$ is cyclic, which intuitively means that the ℓ-isogeny walk should not backtrack. Special soundness implies that the protocol is a proof of knowledge for the hard relation R [16].

To achieve honest-verifier zero-knowledge, we rely on the heuristic assumption that the KLPT variant proposed in [22] does not leak information about $\varphi_{\mathrm{secret}}$. Note that Alice cannot simply output the isogeny $\varphi_{\mathrm{resp}} = \varphi_{\mathrm{chall}} \circ \varphi_{\mathrm{com}} \circ \hat{\varphi}_{\mathrm{secret}}$, since Bob could factor the response, and retrieve $\varphi_{\mathrm{secret}}$.

To construct a signature scheme, the Fiat-Shamir transform is applied to the aforementioned identification protocol [28]. That is, the challenge φ_{chall} is now computed from the result of a hash function H on inputs msg and E_1, and the signature σ is a description of the transcript $(E_1, \varphi_{\mathrm{chall}}, \varphi_{\mathrm{resp}})$.

[3]This may be thought of as a superpower: knowing two endomorphism rings $\mathrm{End}(E)$ and $\mathrm{End}(E')$ allows you to compute an isogeny $E \to E'$, but the caveat is that the degree will be large. In reality, the details are more complex. However, for the goal of this work, it is suitable to think of this as a black box.

2.2.2 1D SQIsign Verification

There are several variants of 1D verification, but on a high level they all use a common framework performing steps equivalent to the following:

1. Compute the codomain of $\varphi_{\mathrm{resp}} : \ E_A \to E_2$.
2. Recompute the hash and compute the codomain of $\varphi_{\mathrm{chall}} : \ E_1 \to E_2'$.
3. Check that E_2 and E_2' are isomorphic and that $\hat{\varphi}_{\mathrm{chall}} \circ \varphi_{\mathrm{resp}}$ is cyclic.

Step 1 is computationally the most expensive, since it involves the computation of the isogeny φ_{resp} of degree 2^e where $e \approx \lceil (15/4) \log_2 p \rceil$, whereas the degree 2^λ of the challenge isogeny in step 2 is solely determined by the security parameter λ.

Verification performance is primarily influenced by two factors: the choice of isogeny encoding, which offers a size-performance trade-off, and the form of the prime p. We begin by examining three isogeny encoding options.

Uncompressed signatures. A rational cyclic isogeny can be described simply by a point that generates its kernel. Hence, our hash function only needs to output a point of order 2^λ, assuming $E[2^\lambda] \subseteq E(\mathbb{F}_{p^2})$. However, the 2^e-isogeny φ_{resp} cannot be efficiently represented by a kernel generator, as such points would live in a large extension field. Instead, we describe φ_{resp} as a sequence of blocks $\varphi_i : E^{(i)} \to E^{(i+1)}$, for which each kernel $\ker \varphi_i$ can be described by a single point $K_i \in E^{(i)}(\mathbb{F}_{p^2})$ of some order which is a power of 2. The largest isogeny whose degree is a power of 2 we can describe by such a rational point K_i is given by the largest f such that $E[2^f] \subseteq E(\mathbb{F}_{p^2})$. For supersingular Montgomery curves, this is given by the largest f such that $2^f \mid p + 1$.

Using x-only arithmetic, a kernel point K can be described by a single $\mathbb{F}_{p^2}$ element. Likewise, a Montgomery curve can be represented only by its A coefficient which is also an $\mathbb{F}_{p^2}$ element. Thus, accounting for a total of 5 kernel points and a curve when p is a 256-bit prime, the total size of the signature is 320 B. We refer to this as the *uncompressed* signature. We refer to this as the *uncompressed* signature.

Compressed signatures. As the previous naming suggests, we can decrease the signature size of σ using *compression.* Instead of a full description of $x(K)$ to define $\varphi : E \to E/\langle K \rangle$, we can exploit the fact that $E[d] \cong \mathbb{Z}_d \times \mathbb{Z}_d$ to write $K = [k_1]P + [k_2]Q$ for a deterministic basis P,Q and describe the isogeny only by the scalars k_1 and k_2. Furthermore, since multiples of K co-prime to d generate the same isogeny, we can always find an equivalent point of the form $K = P + [k]Q$ or $K = [k]P + Q$. Hence, an isogeny can be described by a single scalar $k \in \mathbb{Z}_d$ plus a single bit to specify one of the two forms.

The size of the signature can be further decreased by including a compressed description for $\hat{\varphi}_{\mathrm{chall}}$, which is enough to recover E_1, instead of including E_1 itself. This places an additional burden on the verifier, who now recovers E_1 from the image of $\hat{\varphi}_{\mathrm{chall}}$ but also needs to compute the dual of $\hat{\varphi}_{\mathrm{chall}}$ in order to compare it against the result of the hash. In practice, this is done by picking any point K of order 2^λ such that $[2^{\lambda-1}]K$ is not in the kernel of $\hat{\varphi}_{\mathrm{chall}}$ and computing $\hat{\varphi}_{\mathrm{chall}}(K)$, which is then a kernel generator for the dual of $\hat{\varphi}_{\mathrm{chall}}$. The verifier must then make sure that $\hat{\varphi}_{\mathrm{chall}}(K)$ is equivalent to the output of the hash, meaning that the two points must be scalar multiples of each other. In order to make this check faster in the verification, the aforementioned scalar multiple is computed by the signer and included as part of the signature. This brings the total size of the signature down to 157 B. A detailed analysis of these steps is given in ApresSQI [11, § 4.1].

Despite the tempting improvements in signature size, compression has a high impact on performance since the basis (P,Q) must be computed at each curve $E^{(i)}$ in order to recover the kernel points K_i. This needs to be done through a deterministic method that the signer and verifier have previously agreed upon, and this dominates the cost of the verification in the original NIST proposal. Moreover, as basis sampling is not invariant under isomorphisms, signer and verifier must work on the exact same curve rather than only isomorphic curves. This leads to the concept of *curve normalization*, which the NIST proposal defines as a deterministic method for selecting one curve from an isomorphism class. This procedure is costly, requiring one inversion, one square root and several field multiplications, and was used on both E_1 and E_2.

Seeded signatures. To improve the performance of the basis sampling, ApresSQI revisits an idea from [21] of adding seeds to the signature. Essentially, the P,Q bases are specified in the signature to avoid the cost of sampling them, but they are chosen from within a small pool of candidates so that they can be encoded in just one byte each with negligible failure probability.

While the NIST proposal focused only on compressed signatures, this work analyzes and improves both compressed and uncompressed SQIsign signatures, as we are interested in the fastest verification time possible, possibly at the expense of larger signatures. We do not examine seeded signatures, as improvements in basis sampling have turned it into a negligible part of the cost.

2.2.3 Selecting Primes for Fast 1D Verification

The choice of prime has a significant impact on the verification, since having f as large as possible, with $p = c \cdot 2^f - 1$ allows us to compute the response isogeny in fewer blocks of rational 2^f-isogenies. The original SQIsign [9, 21, 22] could not simply maximize f: For the performance of their KLPT-based signing, the scheme required $p^2 - 1$ to also contain an odd smooth factor $T \mid (p^2 - 1)$ of size approximately $p^{5/4}$, in order for the signer to only need to work with rational isogenies. This condition restricts the search space and led to the prime p_{1973} with $f = 75$, which computes the response isogeny in 13 blocks.

ApresSQI [11] proposed computing the signer's isogenies over larger extension fields, relaxing the constraint $T \mid (p^2 - 1)$ and thus enabling primes with larger f values. Taking advantage of this observation, the authors of [11] proposed to use the prime

$$\mathsf{p}_4 \;=\; 2^{246} \cdot 3 \cdot 67 - 1.$$

In this case $f = 246$, where the response isogeny is computed in just 4 blocks of 2^{246}-isogenies. While the total degree e remains unchanged, fewer blocks benefits both compressed and uncompressed variants greatly: in uncompressed variants, it improves signature size by reducing the number of kernel points that need to be embedded in the signature, and in compressed variants, it improves on performance by requiring fewer basis samplings. On the downside, having to work over large extension fields would likely have a significant impact on signing performance, which was never measured in an optimized implementation.

A significant improvement was recently proposed by Onuki and Nakagawa [42], where instead of working over large extension fields the signer can use 2D isogenies to produce a 1D response isogeny as output. Initial analysis suggests that this work enables practical signing times for primes with large 2^f-torsion, perhaps twice as fast as in classical SQIsign for primes with $f = 75$. This method no longer requires the prime to be related to T in any way, and even lifts the smoothness restriction from it, leaving a large f as the sole criterion for efficiency. In view of this, and in order to provide a more direct comparison with 2D verification techniques, throughout this work we drop ApresSQI's p_4 and instead adopt the same prime used in [3], which has the largest f possible for level-1-sized prime,

$$\mathsf{p}_{248} = 2^{248} \cdot 5 - 1.$$

2.2.4 Higher-dimensional SQIsign

The spectacular break of SIDH/SIKE [8, 38, 46] led to several new techniques using higher-dimensional isogenies in isogeny-based cryptography. In particular, by embedding the signature isogeny inside an isogeny of higher dimension, SQIsignHD [17] avoids very slow parts of the signing procedures of 1D SQIsign, while also providing improved security, as the distribution of response isogenies achieves a uniform distribution, hence provable zero-knowledge on the signature.

SQIsignHD thus embeds φ_{resp} into a 4- or 8-dimensional isogeny between products of elliptic curves. Whereas this greatly improved signing time and security, the downside was that verification was rather slow and maybe even unfeasible in practice, as the computation of 4- or 8-dimensional isogenies is a daunting and complex task.

2D Verification. After Nakagawa and Onuki [40] introduced the algorithm RandIsogImages, and with some inspiration from [43], three separate papers [3, 25, 41], released simultaneously, adapted SQIsign to embed φ_{resp} into a 2D isogeny. These works achieve the same fast signing results and enhanced security from SQIsignHD, while drastically improving on the verification performance thanks to the lower dimension: the C implementation that accompanies SQIsign2D-West requires roughly 9 million cycles [3]. While this result shines in comparison to the 37 million cycles of the 1D verification of the SQIsign NIST proposal [9], so far no work has pushed the 1D verification to its limits. In particular, the improvements from ApresSQI have not made their way to an optimized C implementation. Thus, it is not totally clear if the 2D verification approach is indeed the fastest strategy in *all* aspects.

In the next section, we present a first-order cost model that may help to estimate the relative efficiency of 1D and 2Dl verification approaches.

3 Theoretical Cost of 1D and 2D Verification

Given the algorithmic advancements in 1D verification and the impressive performance reported for 2D SQISign verification [3], a natural question arises: which approach is computationally more efficient? Although it is impossible to give a definite answer, we can give estimates for the computational cost of the pure isogeny computations in both variants using state-of-the-art methods, which dominates the cost of verification in both variants. Theoretically, the cost of a highly optimized implementation of verification should not differ much from the pure isogeny cost of computing the response and challenge isogenies for both 1D and 2D SQIsign. Thus, using theoretical models to compute the number of $\mathbb{F}_p$-operations required for these isogenies answers not only the relative question which variant is more efficient, but also shows how much overhead is left in each variant, that is, how many computations are spent on non-isogeny computations.

Let $\mathbf{m}, \mathbf{s}, \mathbf{a}, \mathbf{i}$ and $\mathbf{M}, \mathbf{S}, \mathbf{A}, \mathbf{I}$ denote the cost of field multiplication, squaring, addition and inversion, over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$, respectively. We will first ignore additions and subtractions, and assume that $\mathbf{m} = \mathbf{s}$, $\mathbf{M} = 3\mathbf{m}$ and $\mathbf{S} = 2\mathbf{m}$, so that we can give a rough estimate in number of $\mathbb{F}_p$-multiplications. The cost of a field inversion may vary significantly depending on the approach used for computing this operation, which we discuss in more detail in Section 5.3. The inversion-to-multiplication cost ratio depends on the security level; based on our experimental results for p_{248}, we use $\mathbf{I} = 163\mathbf{m}$.

We first analyze the optimal cost in terms of finite-field multiplications, before analyzing the optimal cost in terms of cycles. This allows us to additionally account for additions and subtractions, based on the number of cycles each takes in optimal implementations. We do so because additions for the primes used are non-negligible, and 2D isogeny computations use a significant amount of additions. Thus, taking these into account increases the accuracy of the analysis.

3.1 In Terms of $\mathbb{F}_p$-operations

Each variant computes a response and challenge isogeny in verification.

1D. For the 1D variant, the response isogeny consists of a single isogeny of length $\approx 2^{\frac{15}{4}\log p}$. For p_{248}, this can be computed as four isogenies of length 2^f where $f = 248$. The challenge isogeny is of length 2^λ.

The cost of a 2^n isogeny depends on three values: **1)** the cost of computing the codomain of a 4-isogeny, denoted xISOG, at a cost of $4\mathbf{S}$ operations, **2)** the cost of quadrupling a point, by doubling twice, at a cost of $8\mathbf{M} + 4\mathbf{S}$ operations, denoted Q, and **3)** the cost of evaluating the image of a point under a 4-isogeny, denoted xEVAL, at a cost of $6\mathbf{M} + 2\mathbf{S}$ operations [14, 32]. By applying optimal strategies [20, 32], computing a 2^{2e} isogeny as e isogenies of degree 4 always costs the sum of $e \cdot \mathsf{xISOG}$ and a certain number of xEVAL and Q. Different strategies provide a trade-off between the number of xEVAL and Q, with the optimal strategy depending on the ratio of their cost. We apply the formulas from [20] to compute the total cost in terms of $\mathbb{F}_p$-operations.

The cost of xEVAL in terms of $\mathbb{F}_p$-multiplications is $a := 6 \cdot 3 + 2 \cdot 2 = 22$, and for Q we find $b := 8 \cdot 3 + 4 \cdot 2 = 32$. Then, the rough cost of all xEVAL and Q is determined by the root $z_0 \in [0, 1]$ of the polynomial $f(z) = z^a + z^b - 1$, as

$$C_{a,b}(e) := \frac{-1}{\log z_0} \cdot e \log e.$$

With $a = 22$ and $b = 32$, we get $z_0 = 0{,}9743508585$. Then, the cost of a full 2^{2e}-isogeny can be given as

$$C_{\text{iso}}(e) = C_{a,b}(e) + e \cdot \mathsf{xISOG}.$$

By counting the number of $\mathbb{F}_p$-multiplications in our implementation, we find that this theoretical model accurately predicts the cost, being only 2% off. The pure isogeny cost for 1D verification is thus approximately

$$4 \cdot C_{\text{iso}}\left(\frac{f}{2}\right) + C_{\text{iso}}\left(\frac{\lambda}{2}\right) = 4 \cdot 24{,}147 + 10{,}756 = 107{,}344\mathbf{m}.$$

2D. For the 2D variant, the response isogeny consists of a single 2D isogeny of length $\approx 2^{126}$, and a challenge isogeny of length $2^{2\lambda}$.

Similar to the 1D case, the 2D isogeny can be computed using optimal strategies. However, doubling and evaluating have significantly different costs in the first step of the $(2^{126}, 2^{126})$-isogeny compared to later steps, which the optimal strategy needs to take into account [18, § 5.2]. To model the cost, we simplify these steps and compute separate costs for the first and last step, the so-called 'gluing' and 'splitting' isogenies, which we take from Dartois et al. [18]. This underestimates the actual cost of the full isogeny, but only by a few percents. Including further improvements from SQIsign 2D-West, we find $11{,}170\mathbf{m}$ for gluing, and $196\mathbf{m}$ for splitting.

The total cost of the remaining (2,2)-chain of length 125 is then computed similarly as for the 1D case: doubling takes $16\mathbf{M}+16\mathbf{S}$ per step, and evaluating $8\mathbf{M}+8\mathbf{S}$, which gives $a := 40$ and $b := 80$ for a total cost of $50{,}169\mathbf{m}$. We additionally add 124 codomain computation at a cost of $9\mathbf{M} + 8\mathbf{S}$ per xISOG for a total of $5{,}332\mathbf{m}$, bringing the total of the $(2^{126}, 2^{126})$-isogeny to $66{,}867\mathbf{m}$. This model ignores several additional computations, required for theta arithmetic throughout the isogeny computation, and thus again underestimates the actual cost of such a $(2^{126}, 2^{126})$-isogeny. However, we are only a few percent off from the actual cost, counting $\mathbb{F}_p$-operations, in the code of SQIsign2D-West [3].

For the challenge isogeny, we can repeat the computations using the 1D model, with $n = \lambda$, giving $C_{\text{iso}}(\lambda) = 24{,}147\mathbf{m}$ using optimal strategies. Thus, we find a total number of $\mathbb{F}_p$-operations of $91{,}014\mathbf{m}$, somewhat lower than 1D verification.

Taking additions into account. The previous numbers ignore the cost of additions and subtractions. However, for the prime p_{248}, additions are non-negligible, and we experimentally find $\mathbf{a} \approx 0.33\mathbf{m}$ with assembly arithmetic in our Raptor Lake platform. As 2D isogenies use many more additions than 1D isogenies do, we get a more accurate picture when we take these into account.

By including all additions and subtractions in our model, using $\mathbf{a} \approx 0.33\mathbf{m}$, this gives us a rough estimate of the number of $\mathbb{F}_p$-operations required for the pure isogeny computations in each variant: roughly127,000 for the 1D variant, compared to about 116,000 for the 2D variant, where we must note that 2D verification requires several other computations beyond pure isogeny computations, and that our model underestimates the isogeny costs in several places.

Altogether, these numbers suggest that there is no intrinsic advantage towards either variant. This may motivate further research into faster signing algorithms for 1D SQIsign.

Remark 1. The above numbers allow us to compute the remaining "overhead", i.e., non-isogeny, computations in verification. Given the numbers in Table 1, we find that uncompressed 1D verification has very little overhead, whereas 2D verification may potentially still improve: 2D verification *in practice* takes roughly 3 Mcycles more than the estimate given above as it requires a substantial number of additional computations beyond the two pure isogeny costs.

4 Improved Techniques for 1D Verification

In this section, we describe a series of low-level algorithmic improvements and modifications to the signature format that allow us to significantly speed up the verification.

We focus for concreteness on the prime p_{248} for NIST security level 1, but the techniques described can be generalized in a straightforward way to other primes and security levels. We aim to make 1D verification as fast as possible and make no assumptions regarding the algorithmic strategy of the signer, other than the fact that it is able to produce a 1D response isogeny which fits in 4 blocks of rational isogenies for p_{248}.

Given that the cost paid in signature size of using uncompressed points is ameliorated by the smaller number of blocks and that we aim to explore the boundary of 1D verification efficiency, we have carefully studied and optimized both compressed and uncompressed variants. In addition, we present versions of both variants that allow for a near-perfect parallelization of the verification with 5 cores, drastically pushing the performance frontier at the cost of a reasonable increase in signature size.

4.1 Smart-Compressed Signature

We first describe an enhanced version of a compressed SQIsign signature which significantly reduces the performance cost of using point compression by exploiting the *smart sampling* technique introduced in ApresSQI [11], as well as other newly developed techniques that build up on it. We begin by refining Theorem 2 from [11]:

Lemma 1. *Let E_α be a supersingular curve given by $y^2 = x(x - \alpha)(x - 1/\alpha)$ for some $\alpha \in \mathbb{F}_{p^2}$ and let $P = (x,y) \in E_\alpha(\mathbb{F}_{p^2})$. Let 2^f denote the maximal rational two-power* *torsion of E_α. Let b_0 and b_1 be the quadratic Character's of x and $x-\alpha$, respectively. Then*

$$P \in [2]E_\alpha(\mathbb{F}_{p^2}) \quad \Leftrightarrow \quad b_0 = b_1 = 1.$$

In particular, the order of P is a multiple of 2^f if and only if b_0 and b_1 are not both 1. Moreover, if this is the case, b_0 and b_1 precisely determine the point of order 2 over which P lays:

$$\begin{aligned} P \text{ is above } (0,0) &\Leftrightarrow b_0 = 1, b_1 = -1,\\ P \text{ is above } (\alpha,0) &\Leftrightarrow b_0 = -1, b_1 = 1,\\ P \text{ is above } (1/\alpha,0) &\Leftrightarrow b_0 = -1, b_1 = -1. \end{aligned}$$

Proof. This is a direct consequence of [11, Thm. 2] and [12, 47]: For $P \in E_\alpha(\mathbb{F}_{p^2})$, we have $P \in [2]E_\alpha(\mathbb{F}_{p^2})$ if and only if P has trivial Tate pairings with all $Q \in E_\alpha[2]$. These can be computed as the quadratic characters of x, $x-\alpha$ and $x-1/\alpha$. Non-triviality of any of these three then implies the order of P is divisible by 2^f, with the single trivial quadratic character indicating above which $Q \in E_\alpha[2]$ the point P lies. □

Corollary 1. *In the setting of the above Lemma, if $P = (x,y) \in E_\alpha(\mathbb{F}_{p^2})$ with x a non-square, then P is a point of order a multiple of 2^f that does not lay over (0,0). On the other hand, if $P = (x,y) \in E_\alpha(\mathbb{F}_{p^2})$ where $x = z\alpha$ with z a square but $z-1$ a non-square, then P is a point of order a multiple of 2^f that lays over (0,0).*

Proof. The first case is a direct consequence of Lemma 1. For the second case, note that α is always a square [2, 13] and so $x = z\alpha$ is a square but $x - \alpha = (z-1)\alpha$ is not. □

By precomputing a list of non-squares x_i as well as a list of field elements z_i where z_i is a square but $z_i - 1$ is not, Corollary 1 allows us to significantly reduce the cost of sampling a basis in any curve given its α coefficient, as shown in Algorithm 1.

Algorithm 1 SmartSampling (adapted from [11])

Input: $\alpha \in \mathbb{F}_{p^2}$
Output: An x-only basis P,Q of $E_\alpha[2^f]$ where Q lays over (0,0)
Precomputation: Lists $X,Z \subset \mathbb{F}_{p^2}$ with $Z[i]$ a square and $X[i]$, $1 - Z[i]$ non-squares.

```
1: for $z \in Z$ do
2:     $x \leftarrow z\alpha$
3:     if $(x,\_) \in E_\alpha(\mathbb{F}_{p^2})$ then
4:         $Q = (x,\_)$
5:         break
6:     end if
7: end for
8: for $x \in X$ do
9:     if $(x,\_) \in E_\alpha(\mathbb{F}_{p^2})$ then
10:        $P = (x,\_)$
11:        break
12:    end if
13: end for
14: return $[(p+1)/2^f]P,[(p+1)/2^f]Q$
```

The cost of Algorithm 1 is dominated by an average of just four quadratic character computations to check if a point is on the curve. In contrast, the traditional basis sampling method used in the NIST proposal requires the same quadratic character computations but also an expensive chain of f point doublings at every execution of each loop to check the order of the point.

By using Algorithm 1, basis sampling changes from being the dominating cost of verification to just a negligible overhead. The main drawback is that Algorithm 1 requires that the α coefficient for the curve be previously computed. Obtaining α from the Montgomery coefficient A requires a costly square root computation, and this must be performed at every curve (the commitment curve, the challenge curve, the public-key curve, and the 3 intermediate curves of the response isogeny). We now describe a series of novel tricks that bypass this issue.

Alternative public keys. In order to avoid computing α for the public-key curve, we use α itself as the public key rather than the Montgomery A coefficient. This has no effect on the public key size, since both are $\mathbb{F}_{p^2}$ elements, and recovering $A = -\alpha - 1/\alpha$ can be done projectively at a minimal cost.

Slightly shorter isogenies. When computing an isogeny of degree 2^f and pushing the kernel point through each 4-isogeny, before the very last step we are left with a point of order 2 which is always different than (0,0) when using the widespread isogeny formulas of [20]. The x-coordinate of this point is precisely a non-zero root of the current curve polynomial $x(x-\alpha)(x-1/\alpha)$, and so we automatically recover α for this curve. Therefore, it is significantly cheaper to use this point to sample a new basis rather than completing the last step of the isogeny. This means that both signer and verifier must agree to use blocks of 2^{f-1}-isogenies instead of using the entire 2^f torsion. Losing a bit of the 2^f torsion does not present a drawback, as we can still compute the response isogeny for the prime p_{248} using four 2^{f-1}-isogeny blocks.

Smart hashing. The NIST proposal also samples for a basis (P,Q) in the commitment curve since the challenge isogeny is obtained from the kernel point $P + kQ$ where k is the output of the hash. We instead use a novel method based on Corollary 1 to hash directly to a kernel point, thus removing the need to sample for a basis and computing the three-point ladder for $P + kQ$. By using rejection sampling to hash into a non-square which is the x-coordinate of a point in the curve, we are guaranteed to obtain a point which does not lay over (0,0) and whose order is a multiple of 2^f. After multiplying out the cofactor, we obtain a point of order 2^λ that can be used as the kernel point. Even though the output never lays over (0,0) we still have a large enough challenge space, since the number of 2^λ-isogenies whose kernel doesn't contain (0,0) is precisely 2^λ. Using rejection sampling means that the hash is variable time, yet still constant time, as this has no bearing since its inputs (message and commitment) are public. The algorithm is presented as Algorithm 2.

Algorithm 2 SmartHashing

```
Input: A commitment curve $E_1$ and a message msg
Output: An $x$-only point $K \in E_1[2^\lambda]$ not over (0,0)
Note: Uses a hash function $H : \{0,1\}^* \to \mathbb{F}_{p^2}$ and a fixed non-square $\delta \in \mathbb{F}_{p^2}$
 1: ctr ← 0
 2: while 1 do
 3:     $x \leftarrow H(E_1 || \mathsf{msg} || \mathsf{ctr})$
 4:     $x \leftarrow \delta \cdot x^2$
 5:     if $(x,\_) \in E_1(\mathbb{F}_{p^2})$ then
 6:         $K \leftarrow (x,\_)$
 7:         break
 8:     end if
 9:     ctr ← ctr + 1
10: end while
11: return $[(p+1)/2^\lambda]K$
```

By using α as the public key, shorter isogenies and smart hashing, the α coefficients are always readily available and we essentially remove the cost of basis sampling completely.

Unnormalized challenge curve. Curve normalization is another costly procedure that is used in verification, which requires a square root and an inversion. While normalizing the commitment curve is essential in order to recover the same challenge isogeny from the hash, we find, contrary to the NIST proposal, that there is no need to normalize the challenge curve. This was presumed necessary in order to compress the kernel of $\hat{\varphi}_{\text{chall}}$, but instead of using the normalized version of E_2 both parties can agree to use any version of E_2 that results from evaluating $\varphi_{\text{resp}} : E_A \to E_2$ with the common formulas.

Using this unnormalized version of E_2 means that the kernel of $\hat{\varphi}_{\text{resp}}$ lays over (0,0), so the cyclicity check is reduced to ensuring that the kernel generator for $\hat{\varphi}_{\text{chall}}$ is of the form $P + kQ$ and not $kP + Q$[4]. Thus, apart from removing the cost of a curve normalization, this trick removes the cost of the cyclicity check and the need for an extra byte in the signature.

Remark 2. We could easily obtain a seeded variant of this signature by including the indices in the lists X,Z from Algorithm 1 as seeds. However, the smart sampling is already so efficient that this did not improve the verification performance significantly. Thus, we have omitted this variant here.

4.2 Computational Cost of Uncompressed Signatures

The uncompressed variant proposed in ApresSQI removed most of the cost of the verification that is not directly due to isogeny computations: a) kernel points for $\varphi_{\text{resp}} : E_A \to E_2$ are included in the signature explicitly so that basis samplings, point differences and 3-point ladders are not needed, and b) the commitment curve E_1 is included in the signature, which avoids the need to normalize it and allows the verifier to compute $\varphi_{\text{chall}} : E_1 \to E_2'$ in the natural way, without having to compute a generator for the dual. The verifier then confirms that E_2 and E_2' have the same j-invariant, without needing to normalize either. Moreover, the cyclicity check can be performed by recording the second-to-last j-invariant along the way of both isogenies, at a minimal cost, and confirming that they are different.

Previously, the most significant non-isogeny overhead was recovering φ_{chall} from the hash, which still required a basis sampling, point-difference computation, and a three-point ladder. However, by using SmartHashing, we eliminate this overhead entirely, replacing it with an average of just two efficiently optimized quadratic residue computations (see Section 5.3). Hence, the total cost of verification for uncompressed signatures has only negligible overhead over the core cost of one 2^{λ}-isogeny and four 2^f-isogenies.

4.3 Parallelization-friendly Signatures

The 1D verification of SQIsign, which evaluates multiple blocks of 2^f-isogenies using identical instructions, presents a natural opportunity for parallelization. By including the intermediate curves $E^{(i)}$ of the response isogeny $\varphi_{\text{resp}} : E_A \to E^{(1)} \to E^{(2)} \to E^{(3)} \to E_2$, each block $E^{(i)} \to E^{(i+1)'}$ can be computed simultaneously and the verifier only needs to check that the j-invariants of $E^{(i+1)'}$ and $E^{(i+1)}$ match at each step. This can be taken further by using an additional core to also compute the challenge isogeny (or its dual) in parallel. We present simple implementations of this idea using OpenMP, for both the smart-compressed and uncompressed variants.

Remark 3. 2D verification is performed in a single block, so that the above approach used to parallelize 1D verification does not immediately generalize. Future research may consider splitting the 2D isogeny into smaller blocks to allow parallelization, although the intermediate information required for such an approach at first sight seems more costly than in the 1D case.

A naive implementation of the idea using p_{248} would require including 4 more curves in the signature, adding a hefty 256 B. However, by inverting the directions in which

[4]Recall that our SmartSampling always returns the basis point Q over (0,0).

Compressed:

$$\begin{tikzcd} E_A \arrow[r, "\varphi_1"'] & E^{(1)} & E^{(2)} \arrow[l, "\hat{\varphi}_2"] \arrow[r, "\varphi_3"'] & E^{(3)} & E_2 \arrow[l, "\hat{\varphi}_4"] \arrow[r, "\hat{\varphi}_{\text{chall}}"'] & E_1 \end{tikzcd}$$

Uncompressed:

$$\begin{tikzcd} E_A & E^{(1)} \arrow[l, "\hat{\varphi}_1"] \arrow[r, "\varphi_2"'] & E^{(2)} & E^{(3)} \arrow[l, "\hat{\varphi}_3"] \arrow[r, "\varphi_4"'] & E_2 & E_1 \arrow[l, "\varphi_{\text{chall}}"] \end{tikzcd}$$

Figure 2: Direction in which isogenies in the parallel verification are evaluated for the compressed version (top) and uncompressed version (bottom).

some isogenies are computed and providing kernels for the duals instead, we only need to include every other curve in the signature. Using the arrangement shown in Figure 2, the overhead in signature size is reduced by a factor two.

The two variants compute their isogenies in opposite directions as a consequence of staying true to their original objectives: in the compressed version, we only need to include two curves in the signature ($E^{(2)}$ and E_2) which minimizes its size, but verifying the hash still requires computing a dual for $\hat{\varphi}_{\text{chall}}$ and normalizing E_1. On the other hand, the uncompressed version computes $\varphi_{\text{chall}} : E_1 \to E_2$ in the natural way and makes essentially only isogeny computations, but has slightly bigger overhead in signature size since it needs to include three curves ($E^{(1)}$, $E^{(3)}$ and E_1).

Although not strictly necessary, our implementation requires the signer to present the isogenies in a way that none of the kernels include the point (0,0). This is easy to do by applying an inexpensive isomorphism to the intermediate curves and kernel points when needed, and avoids having to add an additional bit per step in the compressed version.

Remark 4. The computation for the 2^λ-isogeny can be embedded into that of a larger 2^f-isogeny that performs dummy calculations for the additional steps. Combined with the fact that we do not need to worry about special cases for isogenies with (0,0) in the kernel, this means that the entirety of the parallel uncompressed variant verification is compliant with a SIMD (single instruction, multiple data) paradigm and would fit a vectorized implementation nicely. The compressed version could also be largely vectorized, but faces three technicalities. First, we must push a point through the dual of the challenge isogeny to get the dual, while the others do not. Second, we need to perform a variable-time basis sampling for each step. And third, the isogeny φ_1 starts from E_A, which we are not free to map through an isomorphism as it is the public key, so it still needs a branch in case (0,0) is in the kernel.

5 A New SQIsign Library for 1D Verification

In this section, we describe our new SQIsign library. It is a modification of the NIST submission's reference library[5] to include our improvements. Namely, we ported it to 32-bit architectures and optimized the verification of 1D SQIsign variants.

The reference library implements SQIsign for NIST security levels 1, 3 and 5. This implementation is comprised of several modules and has GMP as a dependency. The finite-field arithmetic module offers two options: a reference C implementation for all security levels and an optimized assembly implementation for security level 1. The reference implementation is obtained with the Fiat Cryptography [26] code generator.

To add support for Cortex-M4, we created a 32-bit version of the reference library, replaced the dependency on GMP with mini-GMP, and added an option to enable verifi-

[5]Available at `https://github.com/SQISign/the-sqisign`.

cation only (see Section 5.1). Optimization of verification entailed improving finite-field arithmetic and implementing the techniques discussed in Section 4. Regarding finite-field arithmetic, we first incorporated a more efficient portable implementation and an optimized ARMv7E-M assembly implementation (see Section 5.2); then, we improved computation of quadratic characters, inverses and square roots (see Section 5.3).

In addition to implementing the optimized verification variants for p_{248}, our library also implements the entire protocol for the original parameter sets, which benefit from arithmetic and algorithmic improvements where possible, although limited to retain compatibility with the original test vectors. The library is available at:

`https://github.com/Crypto-TII/the-sqisign-1d`

Although the library does not implement any signing variant for p_{248}, it includes valid test vectors that have been generating using a fork of the ApresSQI Sage implementation that we modified to fit the new signature formats. This should be thought of only as a KAT generator and not a reference for implementing the signature, as it still uses ApresSQI's large-extension techniques which are no longer state of the art. This fork is available at:

`https://github.com/Crypto-TII/the-sqisign-1d-apressqi-genkat`

5.1 Cortex-M4 Compatibility

Initial work on the library consisted in modifying the SQIsign implementation submitted to NIST to make it compatible with deeply embedded devices, and in particular the ARM Cortex-M4 core, based on the 32-bit ARMv7E-M architecture. Such devices stand to benefit from the short signature sizes of SQIsign, but their limited computing power, due to slower clock speeds, unavailability of 64-bit arithmetic instructions and lack of superscalar execution capabilities, makes it challenging to run SQIsign operations at an acceptable speed. The improvements described in this subsection bridge much of this gap, although we additionally invested some effort in speeding up the $\mathbb{F}_p$ arithmetic implementation, covered in Section 5.2.2, to further advance toward practical runtimes on this platform.

A first roadblock is that, while the original SQIsign implementation did consider 32-bit architectures, it does not actually compile and run in 32-bit mode. We started by rerunning the Fiat-Crypto code generator [26] to output radix-2^{32} $\mathbb{F}_p$ arithmetic, as the original radix-2^{64} code requires 128-bit integer types which are not available on 32-bit architectures. Several bugs also needed to be fixed, such as in routines that access individual bits of a multiprecision integer but implicitly assumed 64-bit words.

Remark 5. Our library actually contains a full 32-bit port of SQIsign, including the key generation and signing operations. This required further changes to e.g. the `intbig` module which was hardcoded to 64 bits, and including an option to replace the heavyweight GMP library dependency with the much lighter mini-GMP library. However, further work remains to be done for key generation and signing to work on embedded devices; other than the obvious performance issues, the main roadblocks are a very large memory footprint and widespread use of dynamic memory allocation.

Given Remark 5, we opted to implement a verification-only build option, removing unnecessary dependencies related to the key generation- and signing-specific modules of SQIsign, such as `intbig`, `klpt` and `quaternion`. This considerably reduced the number of source files that need to be included in a verification-only build. Additionally, some arrays used to store kernel-point data, whose size grows with the square root of the largest prime degree used for isogenies in signing, were reduced for the verification which uses degree at most 3, significantly reducing the memory footprint. As a result, verification for all SQIsign security levels now fits in devices with as little as 32 KB of RAM, or even 8 KB in some cases.

A characteristic of deeply embedded systems is that they lack virtual memory features found on larger systems. This, combined with the small amount of RAM available (in the range of tens to hundreds of KB), means that using dynamic memory allocation, such as the C standard library's `malloc()` function, is not advised, and indeed strictly forbidden by certain coding standards targeting deeply embedded systems. Thus, we removed a few dynamic memory allocations from the verification-only build. As per Remark 5, much more work remains to be done to remove such allocations from the key generation and signing operations, particularly in the `intbig` module, due to its use of the GMP library.

In order to benchmark SQIsign verification performance on the Cortex-M4, we opted to integrate our library with pqm4 [33]. This presented yet another roadblock, as pqm4 assumes that all operations (key generation, signing and verification) are available. To work around this, we generated fixed key pairs and signatures, and replaced the NIST API functions `crypto_keypair` and `crypto_sign` with mock implementations that just return these fixed key pairs and signature. Verification, of course, is computed normally. This version of our library with pqm4 integration is available at:

https://github.com/Crypto-TII/the-sqisign-1d-pqm4

5.2 Optimizing Finite-Field Arithmetic

As isogeny computations comprise most of the execution time of verification, and ultimately perform arithmetic in $\mathbb{F}_p$, it is clear that faster $\mathbb{F}_p$ arithmetic directly translates into improved verification performance. Thus, we have investigated techniques to speed up portable $\mathbb{F}_p$ arithmetic in Section 5.2.1, as well as targeting deeply embedded platforms based on the ARMv7E-M architecture, and specifically the Cortex-M4 core, in Section 5.2.2.

Remark 6. The SQIsign implementation submitted to NIST also includes an x86-64 assembly implementation of $\mathbb{F}_p$ arithmetic for the level 1 prime p_{1973}, targeting the Intel Broadwell (5th generation Core) microarchitecture. We have adapted this implementation to p_{248}, but note that it could be further improved by considering the larger power-of-two cofactor 2^{248} of $\mathsf{p}_{248} + 1$ (see Section 5.2.1).

5.2.1 Alternative Portable C Arithmetic

The SQIsign reference library synthesizes portable C code for $\mathbb{F}_p$ arithmetic using Fiat-Crypto [26], whose authors claimed at the time to be the "fastest-known C code" for this task. This performance was evaluated in conventional elliptic-curve cryptography scenarios using prime sizes in the same range as those of SQIsign. More recently, Scott [49] introduced a new code generator using quite different design choices compared to Fiat-Crypto. In particular, Fiat-Crypto employs a *saturated* representation, in which a multiprecision integer is split into *limbs* with size matching the CPU word length (i.e. radix 2^{32} or 2^{64} for a 32- or 64-bit CPU, respectively), whereas Scott's generator opts for an *unsaturated* representation, using radix 2^r with $r < 32$ (for 32-bit CPUs) or $r < 64$ (for 64-bit CPUs). Thus, we investigated the use of Scott's code generator to replace Fiat-Crypto for the portable $\mathbb{F}_p$ arithmetic.

We implemented the required changes in our library to support Scott's code generator and performed benchmarks on both Intel and the Cortex-M4. For Intel, we noticed that Scott's generator performs consistently at the protocol level whether `gcc` or `clang` compilers are used, while Fiat-Crypto performs much better with `clang` compared to `gcc`, and indeed generally outperforms Scott's code generator on `clang`. For the Cortex-M4, to the best of our knowledge, pqm4 is only compatible with `gcc` and, in our experience, `gcc` generally produces faster code than `clang` for this platform. As such, we saw significant speedups from the use of Scott's generator in the Cortex-M4, as discussed in Section 6.

5.2.2 Optimized ARMv7E-M Assembly Arithmetic

While Intel high-performance CPUs feature superscalar and out-of-order execution, cores based on the ARMv7E-M architecture are targeted at microcontroller units in embedded applications, with tight constraints on cost (i.e., silicon area) and power consumption. Meeting these constraints requires much simpler single-issue in-order cores, such as the widely-used Cortex-M4.

The performance of single-issue cores correlates quite well with instruction count; not only arithmetic ones, but also loads and stores, as they lack the superscalar out-of-order machinery that excels at hiding memory-access latencies. Register pressure becomes a serious issue, as every spill and reload must be accounted for, which isn't helped in ARMv7E-M by having only fourteen 32-bit general-purpose integer registers, which is insufficient to hold even the two 256-bit inputs to an arithmetic routine. Even worse, while arithmetic and logical instructions have single-cycle latency, loads (as well as some stores) usually cost 2 cycles unless pipelined.

The Cortex-M4 enhanced DSP instruction set extension includes instructions quite suitable for $\mathbb{F}_p$ arithmetic, especially in saturated representation, such as UMULL, UMLAL and UMAAL. All of these perform $32 \times 32 \to 64$-bit multiplications, with the latter two accumulating with either a 64-bit operand (UMLAL) or two 32-bit operands (UMAAL). There is a rich literature of Cortex-M4 implementations exploiting these instructions to achieve excellent performance [1, 29, 50].

We were graciously given access to a multiprecision arithmetic code generator targeting the Cortex-M4.[6] The generator outputs C files with an API compatible with Scott's code generator [49]. It employs a saturated representation and enforces generation of UMULL and UMAAL instructions by the use of preprocessor macros to emit the desired instruction using inline assembly. Some routines seeking higher control over instruction selection and scheduling, as well as register allocation, employ large inline assembly blocks (dozens of instructions).

Multiplication and squaring are performed separately from reduction. For multiplication, a novel hybrid strategy was adopted, by dividing the multiplication rhombus into blocks of four rows. In a 256-bit multiplication with 8 limbs ($A = A_0, \ldots, A_7$ and $B = B_0, \ldots, B_7$), two blocks are required: the first computes $A \times (B_0, B_1, B_2, B_3)$ and the second computes $A \times (B_4, B_5, B_6, B_7)$. Partial results are stored in the result array, and registers are reloaded between blocks. For squaring, the same strategy was applied, additionally optimizing it by eliminating unnecessary multiplications in the lower part of the rhombus and using doubling operations on the partial results from the upper part.

Montgomery reduction is sped up, as in Scott's generator, by exploiting large 2^k cofactors of $p \pm 1$. Primes with this characteristic are often referred to as *Montgomery-friendly primes*; we refer to [27] for further details on this technique, which appears to have been overlooked by the Fiat-Crypto authors. A second source of speedup for the M4 and Scott's code generator (also overlooked by Fiat-Crypto) is avoiding the final conditional subtraction in Montgomery reduction, which is possible if the Montgomery parameter R is chosen so that $R > 4p$ [7, Chapter 2]. The primes p_{1973}, p_4 and p_{248} have bit length at most 254, thus the "ideal" choice $R = 2^{256}$ is applicable for saturated representation.

We used this generator to produce code for all primes found in the original SQIsign submission, as well as p_{248}. The implementation of modular reduction for p_{248} was tweaked from the originally generated code; the high degree of Montgomery-friendliness of this prime allowed us to write an especially compact and performant implementation.

[6] Manuscript under preparation by its authors.

5.3 Other Optimized Field Operations

For the remaining operations in $\mathbb{F}_p$, SQIsign originally implements exponentiation to $(p-3) \bmod 4$ in constant time through an addition chain for primes of form $p \equiv 3 \bmod 4$. The result is then reused to compute inversions, Legendre symbols and square roots in $\mathbb{F}_p$ by adjusting the result to the correct powers (respectively $p-2$, $\frac{p-1}{2}$, $\frac{p+1}{4}$). We extended the library to implement faster alternatives. For inversion, we adapted the Montgomery version [31] of the constant-time Bernstein-Yang (BY) algorithm [4], such that it can be reused for signing. For Legendre symbols, we implemented a variable-time version of Pornin's algorithm [45] ported from the Rust version available in [54]. Pornin's algorithm is actually simpler and faster to implement in variable-time than a BY-based variant [30].

In SQIsign's NIST submission source code, inverting in $\mathbb{F}_{p^2}$ amounts to computing an inversion, two multiplications, two squarings, and a few additions in $\mathbb{F}_p$. Quadratic character computations in $\mathbb{F}_{p^2}$ require two squarings, one addition, and a Legendre symbol in $\mathbb{F}_p$. The computation of square roots in $\mathbb{F}_{p^2}$ requires two square roots and two inversions in $\mathbb{F}_p$ (although one of them consists of just inverting 2, which can be precomputed), for a total of four exponentiations. In this work, we employ Scott's fast square root method [48], reducing this to just two exponentiation computations in $\mathbb{F}_p$, without any precomputation through a novel reformulation reported in Algorithm 3, applicable only to primes $p \equiv 3 \bmod 4$.

Algorithm 3 Deterministic square roots in $\mathbb{F}_{p^2}$, for $p \equiv 3 \bmod 4$

Input: A quadratic residue $a = a_0 + ia_1 \in \mathbb{F}_{p^2}$, with $i^2 = -1$
Output: A deterministic $x \in \mathbb{F}_{p^2}$ such that $x^2 = a$

```
1: $\delta \leftarrow (a_0^2 + a_1^2)^{(p+1)/4}$
2: $x_0 \leftarrow a_0 + \delta$
3: $t_0 \leftarrow 2x_0$
4: $x_1 \leftarrow t_0^{(p-3)/4}$
5: $x_0 \leftarrow x_0 x_1$
6: $x_1 \leftarrow a_1 x_1$
7: $t_1 \leftarrow (2x_0)^2$
8: if $t_1 = t_0$ then
9:     return $x_0 + ix_1$
10: else
11:     return $x_1 - ix_0$
12: end if
```

6 Results and Conclusion

pqm4 Results. We display Cortex-M4 results in Table 2, benchmarked using the pqm4 framework [33] on ST Microelectronics' NUCLEO-L4R5ZI board. The compiler is `gcc` 13.2.1. As usual, the core runs at a reduced clock (24 MHz) to avoid the use of wait states for Flash memory.

To isolate the effect of our algorithmic improvements, we prepared a version of the library which makes minimal changes with respect to the NIST implementation, only to make it work in 32 bits but without any further improvements. With respect to this baseline for verification, our algorithmic improvements achieve speedups of 3.88×, 5.84× and 8.52× for security levels 1, 3 and 5, respectively, compared to the NIST submission ported to 32-bit architectures. For the new parameter set, the smart sampling variant of Section 4.1 further improves this to 15× for level 1, combined with a reduction in signature size. Trading off signature size for improved performance, a speedup of 22.5× is achievable with the uncompressed variant of Section 4.2. Scaling to the CPU's nominal 120 MHz clock speed[7], we reach execution times of ≈ 1 second and ≈ 0.7 seconds for smart and uncompressed variants, respectively.

[7]Some performance losses are expected, as Flash memory wait states are required for clock speeds above 30 MHz, although most of it should be mitigated by ST's ART accelerator Flash caching subsystem.

Table 2: Cortex-M4 performance, code size and RAM usage results. Sizes reported in KB. Speedup relative to the performance of the NIST submission at the same security level for verification.

Sec. level	Prime	Implementation	$\mathbb{F}_p$ arith.	Performance (Mcycles)	Speedup	Code size	RAM usage
1	p_{1973}	NIST subm.	Fiat	1849	1.00×	229	12.8
		Ours	Fiat	1176	1.57×	111	8.68
			§5.2.1	757.7	2.44×	85.8	9.14
			§5.2.2	477.1	3.88×	89.7	8.58
	p_{248}	Ours, smart	Fiat	391.3	4.73×	86.1	4.47
			§5.2.1	170.7	10.8×	60.7	4.96
			§5.2.2	123.4	15.0×	62.4	4.45
		Ours, uncomp.	Fiat	260.3	7.10×	86.4	5.00
			§5.2.1	114.2	16.2×	61.0	5.40
			§5.2.2	82.3	22.5×	62.8	4.89
3	p_{47441}	NIST subm.	Fiat	9948	1.00×	270	18.3
		Ours	Fiat	4600	2.16×	150	12.8
			§5.2.1	2670	3.73×	106	14.0
			§5.2.2	1704	5.84×	104	12.9
5	p_{318233}	NIST subm.	Fiat	31204	1.00×	326	24.0
		Ours	Fiat	11230	2.78×	202	16.8
			§5.2.1	5387	5.79×	129	17.9
			§5.2.2	3660	8.52×	127	16.7

Our implementation also reduces code size significantly, mostly by not compiling unnecessary code as a result of the verification-only build option. The improved $\mathbb{F}_p$ arithmetic implementations of Sections 5.2.1 and 5.2.2 further reduce code size significantly, in addition to large performance benefits. The RAM usage figures for the baseline implementation incorporate the memory optimizations discussed in Section 5.1.[8] Our library achieves a welcome reduction in RAM usage when using the same Fiat Cryptography code generator as the NIST submission, and the M4-optimized $\mathbb{F}_p$ arithmetic has a negligible effect in this metric. We highlight the figures for the smart sampling variant (62.4 KB of code, 4.45 KB of RAM), which are eminently practical for many applications.

1D verification in high-end processors. We have furthermore conducted benchmarks on high-performance processors which are directly comparable to previous results. We compare our library with the SQIsign NIST submission implementation, as well as the 2D implementation of SQIsign 2D-West[9] [3], with the results shown in Table 1. All libraries use the same reference implementation for finite-field arithmetic (Fiat Cryptography), so columns 4 and 5 allow for a direct comparison. The optimized finite field arithmetic for SQIsign 2D-West incorporates intrinsics for addition, subtraction and multiplication instructions on Intel; while for NIST SQIsign and our library, the optimized version is assembly code targeting the Broadwell architecture. While different in nature, both are architecture-specific and our own micro-benchmarks show that their performance at the

[8]Prior to these optimizations, some of the security levels did not even fit in our target development board; level 5 in particular required close to a megabyte of RAM.

[9]At time of writing, the source code is not publicly available, but the authors provided us with access to it and permission to benchmark it.

field-arithmetic level is roughly equal, hence columns 6 and 7 of Table 1 offer a roughly fair comparison. The results of our micro-benchmarks are in Table 3. All code was compiled on the same Linux-based system using clang 18.1.3 with flags `-march=native -O3` and linking dynamically with GMP (ver. 6.3). The benchmarks were executed on a 13th Gen Intel Core i7-13700K (Raptor Lake) processor with TurboBoost and hyperthreading disabled.

Table 3: Field arithmetic performance for p_{248}.

Field	Arithmetic	Add	Sub	Mul	Sqr	Inv	IsSquare	Sqrt
$\mathbb{F}_p$	Ours, reference	18	13	74	69	12151	6860	21746
	Ours, optimized	9	9	42	41	9399	5918	15766
	2D-West, optimized	13	13	37	27	7141	6760	6809
$\mathbb{F}_{p^2}$	Ours, reference	37	26	320	205	12650	7068	48545
	Ours, optimized	24	22	132	90	9639	6032	32659
	2D-West, optimized	17	18	151	84	7263	6831	28149

With almost the entirety of the running time dedicated to the 2-isogeny chains in our uncompressed variant, improvements are most likely in arithmetic or an algorithmic breakthrough that could reduce the degree of the response isogeny. As things stand, we believe our results are close to the fastest possible sequential 1D verification for response isogenies of the current length.

Concluding remarks. Overall, our results suggest that 1D SQIsign verification can be fully competitive with 2D verification, with even our most size-efficient variant performing on par with 2D-West [3] and the uncompressed variant providing close to a 50% speed-up. On top of that, the inherent parallelizability of 1D verification allows it to be up to 560% faster at a modest increase in signature size. We have exploited the flexibility in trade-offs that 1D enjoys to report record-breaking performance for SQIsign verification and, for the first time, to bring SQIsign to the realm of embedded devices in practice. Although a 2D verification can in principle apply similar techniques, including the required additional data leads to a trade-off that, on first sight, seems more punishing for the signature size.

We stress that our analysis focuses solely on verification and we must concede the current superiority of 2D variants in terms of signing performance. Nevertheless, results such as [42], exploiting 2D techniques in signing while keeping the verification 1D, have the potential to put the spotlight back on a 1D verification, and we hope our encouraging results will motivate further research in this direction.

Lastly, we believe that there are considerable speedups to be attained in implementations using AVX-512 vector instructions, similar to speedups obtained for SIDH [10], and NEON instructions for ARMv8-A [35, 51]. Exploring such an implementation is exciting future work.

Optimized assembly arithmetic for p_{248}. Most recently, Longa [36] provided us access to optimized finite-field arithmetic specialized to the prime p_{248}, which is based on work from [37] and roughly 20% faster than the finite-field arithmetic used in this work. The benchmarks using this arithmetic for 1D SQIsign verification improve to 6.94 Mcycles for compressed and 4.56 Mcycles for uncompressed without parallelization, down to 1.66 Mcycles for compressed and 1.09 Mcycles for uncompressed with parallelization. Similar speedups are expected for 2D-West.

References

[1] Mila Anastasova, Reza Azarderakhsh, and Mehran Mozaffari Kermani. "Fast Strategies for the Implementation of SIKE Round 3 on ARM Cortex-M4". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.10 (2021), pp. 4129–4141. DOI: 10.1109/TCSI.2021.3096916.

[2] Roland Auer and Jaap Top. "Legendre Elliptic Curves over Finite Fields". In: *Journal of Number Theory* 95.2 (2002), pp. 303–312. ISSN: 0022-314X. DOI: https://doi.org/10.1006/jnth.2001.2760. URL: https://www.sciencedirect.com/science/article/pii/S0022314X0192760X.

[3] Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. *SQIsign2D-West: The Fast, the Small, and the Safer*. Cryptology ePrint Archive, Paper 2024/760. https://eprint.iacr.org/2024/760. 2024. URL: https://eprint.iacr.org/2024/760.

[4] Daniel J. Bernstein and Bo-Yin Yang. "Fast constant-time gcd computation and modular inversion". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.3 (2019), pp. 340–398.

[5] Ward Beullens, Luca De Feo, Steven D. Galbraith, and Christophe Petit. "Proving knowledge of isogenies: a survey". In: *Des. Codes Cryptogr.* 91.11 (2023), pp. 3425–3456.

[6] Jean-François Biasse, David Jao, and Anirudh Sankar. "A Quantum Algorithm for Computing Isogenies between Supersingular Elliptic Curves". In: *INDOCRYPT*. Vol. 8885. Lecture Notes in Computer Science. Springer, 2014, pp. 428–442.

[7] J.W. Bos and A.K. Lenstra. *Topics in Computational Number Theory Inspired by Peter L. Montgomery*. London Mathematical Society lecture note series. Cambridge University Press, 2017. ISBN: 9781107109353. URL: https://books.google.com.br/books?id=6BkODwAAQBAJ.

[8] Wouter Castryck and Thomas Decru. "An Efficient Key Recovery Attack on SIDH". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 423–447. DOI: 10.1007/978-3-031-30589-4_15. URL: https://doi.org/10.1007/978-3-031-30589-4_15.

[9] 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*. National Institute of Standards and Technology. 2023. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/sqisign-spec-web.pdf.

[10] Hao Cheng, Georgios Fotiadis, Johann Groszschädl, and Peter YA Ryan. "Highly vectorized SIKE for AVX-512". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)* 2022.2 (2022).

[11] Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. "AprèsSQI: extra fast verification for SQIsign using extension-field signing". In: *EUROCRYPT '24*. Springer. 2024, pp. 63–93.

[12] Maria Corte-Real Santos and Krijn Reijnders. *Return of the Kummer: a toolbox for genus 2 cryptography*. Cryptology ePrint Archive, Paper 2024/948. https://eprint.iacr.org/2024/948. 2024. URL: https://eprint.iacr.org/2024/948.

[13] Craig Costello. “Computing Supersingular Isogenies on Kummer Surfaces”. In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III.* Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11274. Lecture Notes in Computer Science. Springer, 2018, pp. 428–456. DOI: 10.1007/978-3-030-03332-3_16. URL: https://doi.org/10.1007/978-3-030-03332-3_16.

[14] Craig Costello and Hüseyin Hisil. “A Simple and Compact Algorithm for SIDH with Arbitrary Degree Isogenies”. In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II.* Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10625. Lecture Notes in Computer Science. Springer, 2017, pp. 303–329. DOI: 10.1007/978-3-319-70697-9_11. URL: https://doi.org/10.1007/978-3-319-70697-9_11.

[15] Craig Costello and Benjamin Smith. “Montgomery curves and their arithmetic - The case of large characteristic fields”. In: *J. Cryptogr. Eng.* 8.3 (2018), pp. 227–240.

[16] Ivan Damgård. *On Σ-protocols.* https://www.cs.au.dk/~ivan/Sigma.pdf. v2. 2010.

[17] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. “SQISignHD: New Dimensions in Cryptography”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 436. URL: https://eprint.iacr.org/2023/436.

[18] 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: *IACR Cryptol. ePrint Arch.* (2023), p. 1747. URL: https://eprint.iacr.org/2023/1747.

[19] Luca De Feo. “Mathematics of Isogeny Based Cryptography”. In: *CoRR* abs/1711.04062 (2017). arXiv: 1711.04062. URL: http://arxiv.org/abs/1711.04062.

[20] Luca De Feo, David Jao, and Jérôme Plût. “Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies”. In: *J. Math. Cryptol.* 8.3 (2014), pp. 209–247. DOI: 10.1515/jmc-2012-0015. URL: https://doi.org/10.1515/jmc-2012-0015.

[21] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. “SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies”. In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I.* Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. Lecture Notes in Computer Science. Springer, 2020, pp. 64–93. DOI: 10.1007/978-3-030-64837-4_3. URL: https://doi.org/10.1007/978-3-030-64837-4_3.

[22] Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. “New Algorithms for the Deuring Correspondence - Towards Practical and Secure SQISign Signatures”. In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V.* Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 659–690. DOI: 10.1007/978-3-031-30589-4_23. URL: https://doi.org/10.1007/978-3-031-30589-4_23.

[23] Christina Delfs and Steven D. Galbraith. “Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$”. In: *Des. Codes Cryptogr.* 78.2 (2016), pp. 425–440.

[24] Max Deuring. “Die Typen der Multiplikatorenringe elliptischer Funktionenkörper”. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 14 (1941), pp. 197–272.

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

[26] Andres Erbsen, Jade Philipoom, Jason Gross, Robert Sloan, and Adam Chlipala. “Simple High-Level Code for Cryptographic Arithmetic - With Proofs, Without Compromises”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 1202–1219. DOI: 10.1109/SP.2019.00005.

[27] Armando Faz-Hernández, Julio López, Eduardo Ochoa-Jiménez, and Francisco Rodríguez-Henríquez. “A Faster Software Implementation of the Supersingular Isogeny Diffie-Hellman Key Exchange Protocol”. In: *IEEE Transactions on Computers* 67.11 (2018), pp. 1622–1636. DOI: 10.1109/TC.2017.2771535.

[28] 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*. Springer. 1986, pp. 186–194.

[29] H. Fujii and D. F. Aranha. “Curve25519 for the Cortex-M4 and beyond”. In: June 2017. URL: http://www.cs.haifa.ac.il/~orrd/LC17/paper39.pdf.

[30] Mike Hamburg. *Computing the Jacobi symbol using Bernstein-Yang*. Cryptology ePrint Archive, Paper 2021/1271. https://eprint.iacr.org/2021/1271. 2021. URL: https://eprint.iacr.org/2021/1271.

[31] Benjamin Salling Hvass, Diego F. Aranha, and Bas Spitters. “High-Assurance Field Inversion for Curve-Based Cryptography”. In: *CSF*. IEEE, 2023, pp. 552–567.

[32] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. *SIKE*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. National Institute of Standards and Technology, 2022.

[33] Matthias J. Kannwischer, Markus Krausz, Richard Petri, and Shang-Yi Yang. *pqm4: Benchmarking NIST Additional Post-Quantum Signature Schemes on Microcontrollers*. Cryptology ePrint Archive, Paper 2024/112. https://eprint.iacr.org/2024/112. 2024. URL: https://eprint.iacr.org/2024/112.

[34] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. “On the quaternion-isogeny path problem”. In: *LMS Journal of Computation and Mathematics* 17.A (2014), pp. 418–432.

[35] Zhe Liu, Kimmo Järvinen, Weiqiang Liu, and Hwajeong Seo. “Multiprecision Multiplication on ARMv8”. In: *2017 IEEE 24th Symposium on Computer Arithmetic (ARITH)*. 2017, pp. 10–17. DOI: 10.1109/ARITH.2017.27.

[36] Patrick Longa. Personal Communication. 2024.

[37] Patrick Longa. “Efficient algorithms for large prime characteristic fields and their application to bilinear pairings”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2023.3 (2023), pp. 445–472.

[38] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. "A Direct Key Recovery Attack on SIDH". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 448–471. DOI: `10.1007/978-3-031-30589-4\_16`. URL: `https://doi.org/10.1007/978-3-031-30589-4\_16`.

[39] Peter L. Montgomery. "Speeding the Pollard and elliptic curve methods of factorization". In: *Mathematics of Computation* 48 (1987), pp. 243–264.

[40] Kohei Nakagawa and Hiroshi Onuki. "QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras". In: *Cryptology ePrint Archive* (2023).

[41] Kohei Nakagawa and Hiroshi Onuki. *SQIsign2D-East: A New Signature Scheme Using 2-dimensional Isogenies*. Cryptology ePrint Archive, Paper 2024/771. `https://eprint.iacr.org/2024/771`. 2024. URL: `https://eprint.iacr.org/2024/771`.

[42] Hiroshi Onuki and Kohei Nakagawa. *Ideal-to-isogeny algorithm using 2-dimensional isogenies and its application to SQIsign*. Cryptology ePrint Archive, Paper 2024/778. `https://eprint.iacr.org/2024/778`. 2024. URL: `https://eprint.iacr.org/2024/778`.

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

[44] Aurel Page and Benjamin Wesolowski. "The Supersingular Endomorphism Ring and One Endomorphism Problems are Equivalent". In: *EUROCRYPT (6)*. Vol. 14656. Lecture Notes in Computer Science. Springer, 2024, pp. 388–417.

[45] Thomas Pornin. *Faster Modular Inversion and Legendre Symbol, and an X25519 Speed Record*. `https://research.nccgroup.com/2020/09/28/faster-modular-inversion-and-legendre-symbol-and-an-x25519-speed-record/`. 2020.

[46] Damien Robert. "Breaking SIDH in Polynomial Time". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 472–503. DOI: `10.1007/978-3-031-30589-4\_17`. URL: `https://doi.org/10.1007/978-3-031-30589-4\_17`.

[47] Damien Robert. *The geometric interpretation of the Tate pairing and its applications*. Cryptology ePrint Archive, Paper 2023/177. `https://eprint.iacr.org/2023/177`. 2023. URL: `https://eprint.iacr.org/2023/177`.

[48] Michael Scott. *A note on the calculation of some functions in finite fields: Tricks of the Trade*. Cryptology ePrint Archive, Paper 2020/1497. `https://eprint.iacr.org/2020/1497`. 2020. URL: `https://eprint.iacr.org/2020/1497`.

[49] Michael Scott. *Elliptic Curve Cryptography for the masses: Simple and fast finite field arithmetic*. Cryptology ePrint Archive, Paper 2024/779. `https://eprint.iacr.org/2024/779`. 2024. URL: `https://eprint.iacr.org/2024/779`.

[50] Hwajeong Seo and Reza Azarderakhsh. "Curve448 on 32-Bit ARM Cortex-M4". In: *Information Security and Cryptology – ICISC 2020: 23rd International Conference, Seoul, South Korea, December 2–4, 2020, Proceedings*. Springer-Verlag, 2020, 125–139. ISBN: 978-3-030-68889-9.

[51] Hwajeong Seo, Pakize Sanal, Wai-Kong Lee, and Reza Azarderakhsh. "No Silver Bullet: Optimized Montgomery Multiplication on Various 64-Bit ARM Platforms". In: *International Conference on Information Security Applications*. Springer. 2021, pp. 194–205.

[52] National Institute of Standards and Technology (NIST). *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. 2022. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

[53] John Tate. "Endomorphisms of Abelian Varieties over Finite Fields." In: *Inventiones mathematicae* 2 (1966), pp. 134–144.

[54] Alexei Vambol. *A collection of Elliptic Curves for ZkCrypto traits*. https://github.com/privacy-scaling-explorations/halo2curves/pull/95. 2023.

[55] Benjamin Wesolowski. "The supersingular isogeny path and endomorphism ring problems are equivalent". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 1100–1111.