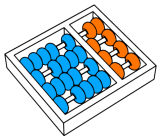

SQISign: A Post-Quantum Signature Scheme

D. Santos *J. López*

Relatório Técnico - IC-PFG-24-09
Projeto Final de Graduação
2024 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

SQISign:
A Post-Quantum Signature Scheme

David Afonso Borges dos Santos* Julio López*

Abstract

This work is a study about the post-quantum signature scheme SQISign, one of the candidates in the NIST Post-Quantum Cryptography Standardization contest. The SQISign algorithm assumes the hardness of finding a path in supersingular isogeny graphs and uses the Deuring correspondence to operate in the quaternion algebra world during signature and in the elliptic curves world during verification. Among the other candidates in the same category, SQISign has relatively small public key and signature sizes, which is an important advantage. The recent SIDH attacks [CD22] showed new ways of efficiently representing isogenies. This fact, resulted in some new variants of SQISign, now using 2, 4, and 8-dimension isogenies. Among the available variants, we are going to discuss SQISign2D-West [Bas+24] and SQISignHD [Dar+23].

1 Introduction

Nowadays, cryptographic algorithms uses go beyond the traditional and most straightforward usage of simply encrypting and decrypting information. They can be used to build up more complex schemes, such as signature schemes, which can be used to prove the identity of a user [FS87].

The algorithms in use today can be divided into two groups, the symmetric algorithms, in which the users share a secret piece of information prior to the use of the algorithm, and the asymmetric algorithms, where such sharing of information is not needed. Usually, asymmetric algorithms are based on mathematical problems, which are considered computationally hard, meaning that if a solution is not available, it is still possible to use algorithms to find a valid solution, but the time necessary using the most optimized algorithms would still be prohibitively long. This happens at the RSA algorithm which assume factoring a large number is computationally hard and at the elliptic curves based algorithms which have an equivalent assumption.

As of today, there has already been developed an algorithm, the Shor's algorithm [Sho99], which can factorize large numbers and compute discrete logarithms in polynomial time provided the existence of a quantum computer. As a result, as soon as a quantum computer is commercially available, the cryptographic algorithms which are based on the problem of factoring a large number or on the problem of computing discrete logarithms and at equivalent problems will be efficiently broken.

Since this possibility only increases over time, it is necessary to find new cryptographic algorithms that would be resistant even against quantum computer attacks. With that in mind, the National Institute of Standards and Technology (NIST) started the process to select, evaluate and standardize post-quantum algorithms. In 2022, NIST selected some algorithms to become standard, among the selected digital signature algorithms, CRYSTALS-DILITHIUM and FALCON are both lattice based algorithms. This dependence on lattices became a reason for an additional

*Instituto de Computação, Universidade Estadual de Campinas (UNICAMP), Campinas, Brazil

call for digital signature algorithms, where the SQISign algorithm is taking part. SQISign is the only isogeny based algorithm in this new call and has public key and signature sizes small when compared to the other signature algorithms in the contest, which is a significant advantage.

In this work, we present a study about the SQISign algorithm. We gather the elements and functioning principles that are necessary to understand how SQISign work. We talk about some SQISign implementations, some of its weaknesses and how new algorithms using isogenies are being built targeting these weaknesses.

In Section 2 there will be a brief introduction to the mathematical fundamentals necessary to understand the algorithm, then in Section 3 there will be a discussion about some of SQISign's building blocks. In Section 4 the SQISign protocol and the key generation, signature and verification will be discussed. The Section 5 talks about the standard implementation submitted to NIST. The Section 6 mentions other implementations of SQISign. Finally, in Section 7 the SQISign algorithm will be compared to some recent SQISign based digital signature schemes.

2 Mathematical Fundamentals

For the signature module, it is necessary some definitions from elliptic curves and quaternion algebra theories and the Deuring Correspondence, which allows establishing a relation between elements in both.

For the verification module, it is only necessary some definitions from the elliptic curves theory and the isogeny path problem in a super singular elliptic curve, which is considered resistant to the Shor's algorithm [Sho99].

2.1 Preliminaries

Firstly, it is necessary some definitions. The definitions below are from [IHH91].

We say that an integer number a divides another integer number b and denote by $a \mid b$ if exists another integer number c, such that $a \times c = b$. If a does not divide b we denote by $a \nmid b$, and there exists an integer number $r < a$ and an integer number d such that $a \times d + r = b$.

We denote by $gcd(a, b) = (a, b) = g$ the greatest integer number g such that g simultaneously divides a and b.

Given integers a, b and c, we say that a and b are congruent module c and denote by $a \equiv b \pmod{c}$ if $c \mid (a - b)$.

We say that a set $S = \{x_1, x_2, ..., x_m\}$ is a complete residue system module m if, for all integers a exists a unique element x in S such that $a \equiv x \pmod{m}$. The set S is said to have cardinality m if it has m elements.

The Chinese Remainder Theorem allows solving simultaneous congruences. For a set of integers $M = \{m_1, m_2, ..., m_n\}$ relatively prime in pairs, and another set $A = \{a_1, a_2, ...a_n\}$ it is possible to find $x \equiv a_i \pmod{m_i}$ for all $i \in [1, n]$.

2.2 Finite Field Arithmetic

A finite set of elements with two operations, addition and multiplication, that satisfied certain properties is called a finite field. Given a prime number, p, the set $\mathbb{F}_p = \{0, 1, ..., p - 1\}$ together with the operations $a + b := a + b \pmod{p}$ and $a \times b := a \times b \pmod{p}$, for a and b in $\mathbb{F}_p$ is a finite field.

For SQISign, specifically, there is interest in finite fields over p^2 where p is a prime number. Since the primes used in SQISign are all in the form $p \equiv 3 \pmod{4}$, an element $c \in \mathbb{F}_{p^2}$ can be represented

by a complex number $c = a + b \times i$ where $a \in \mathbb{F}_p$ and $b \in \mathbb{F}_p$. The operations performed on $\mathbb{F}_{p^2}$ using this representation are similar with usual complex number operations. After performing the complex number operations, however, the residue modulo p is taken from both the real part a and the complex part b of the resulting complex number to get the correspondent field element.

2.3 Elliptic Curves

The SQISign algorithm has special interest in Montgomery Elliptic curves. They can be defined over finite fields $\mathbb{F}_q$ and are represented by equations in the form of

$$By^2 = x^3 + Ax^2 + x, \tag{1}$$

where A and B are field elements with $B(A^2 - 4) \neq 0$.

For SQISign specifically, the curves are defined over fields $\mathbb{F}_{p^2}$. The points that satisfy equation (1) congruent module p^2, together with the point at infinity represented by ∞ is a finite set. It is possible, then, to define operations over the elements of this set. More specifically, taking two distinct points P and Q we can define the operation of point addition using the coordinates of both points. We can also define the operation of point doubling using the coordinates of a point P in the curve.

In order to gain efficiency, it is common to represent elliptic curves using projective coordinates and to perform operations using only the x-coordinate of the points. The equations for point addition and point doubling are explained in detail at [CS17]. The equation of point addition from [CS17], where $X_\oplus$ and $Z_\oplus$ are the coordinates of the point $P + Q$ and $X_\ominus$ and $Z_\ominus$ are the coordinates of the point $P - Q$, is reproduced below

$$\begin{aligned} X_\oplus &= Z_\ominus[(X_P - Z_P)(X_Q + Z_Q) + (X_P + Z_P)(X_Q - Z_Q)]^2, \\ Z_\oplus &= X_\ominus[X_P - Z_P](X_Q + Z_Q) - (X_P + Z_P)(X_Q - Z_Q)]^2, \end{aligned}$$

and the equation of point doubling from [CS17], where X_P and Z_P are the coordinates of the point and A is the A coefficient of the Montgomery Elliptic Curve where the point lies is reproduced below

$$\begin{aligned} X_{[2]P} &= (X_P + Z_P)^2(X_P - Z_P)^2, \\ Z_{[2]P} &= (4X_PZ_P)(X_P - Z_P)^2 + ((A + 2)/4)(4X_PZ_P). \end{aligned}$$

With point addition and point doubling, it is possible to multiply a point by a scalar using a sequence of point additions and point doublings. The method chosen to perform the multiplication by scalar in SQISign is the Montgomery ladder, described also at [CS17]. Finally, the operations of point addition and multiplication by scalar together with the set of points that satisfy equation one is a finite group.

Every elliptic curve has a unique associated value called j-invariant with equation as shown below

$$j(E_{A,B}) = \frac{256(A^2 - 3)^3}{A^2 - 4}. \tag{2}$$

2.4 Isomorphisms

It is possible, then, to create maps between elliptic curves. A non-constant map ϕ between two different elliptic curves E_1 and E_2 such that $j(E_1) = j(E_2)$ is called an isomorphism. Then, there is a unique map ϕ^{-1} called the inverse of ϕ such that $\phi^{-1}(\phi(E_1)) = E_1$.

2.5 Isogenies

If the map ϕ between two different curves E_1 and E_2 is such that $\phi(E_1(\infty)) = E_2(\infty)$ and $j(E_1) \neq j(E_2)$, ϕ is called an isogeny. The set of points $S = \{P \in E_1 | \phi(P) = E2(\infty)\}$ is called the kernel of ϕ.

The isogeny is said to be separable if the degree of the isogeny is equal to the number of elements in its kernel [Feo+22]. We also have that for every finite subgroup G of E there is an associated separable isogeny unique up to isomorphims with kernel G [Feo+22].

For every isogeny ϕ of degree n exists a map ψ, also with degree n, such that $\psi(\phi(E_1))$ is equal to the multiplication by n map in E_1. An isogeny with degree n prime is called an n-isogeny. If n is a composite number, which means there exists a set S of prime numbers such that $n = \prod s_i^{e_i}$, it is possible to build the isogeny ϕ as a composition of smaller degree isogenies, $\phi = \phi_1 \circ \phi_2 \circ ...\phi_i$ with degree $\phi_i = s_i^{e_i}$. This is an important feature for the computational aspect of evaluating isogenies, which will be discussed later.

2.6 Torsion subgroup

In [Cha+23] is defined the m-torsion subgroup of an elliptic curve E, $E[m]$, as the subset of points P in E, such that $[m]P = \infty$. It is also said that there are non-unique points R and S that generate $E[m]$. The SQISign algorithm has routines to find a deterministic basis for such groups.

2.7 Endomorphisms

There is, finally, the case where the map ϕ is from an elliptic curve E to itself, ϕ, in this case, is called an endomorphism. From [Sil09] chapter III.4, we can define operations using endomorphism, where, for two endomorphisms ϕ and Φ the sum is given by

$$(\phi + \Phi)(P) = \phi(P) + \Phi(P),$$

and the multiplication is given by composition

$$(\phi\Phi)(P) = \phi(\Phi(P)).$$

The finite group with the endomorphisms of an elliptic curve E with the operations described above is the endomorphism ring of E.

Then, as stated in [Sil09], for a curve E, the endomorphism ring of E can be equivalent to an order in an imaginary quadratic field or an order in a quaternion algebra. If the latter holds, the curve E is said to be supersingular. Since the SQISign algorithm uses supersingular curves, the definition of a quaternion order will be given later.

In the rest of this document, as in [Cha+23] E_0 will be used to refer to the curve $y^2 = x^3 + x$ which is supersingular since the prime numbers p used in the algorithm are such that $p \equiv 3 \pmod 4$.

2.8 Quaternions

The SQISign uses quaternions to generate keys and generate signatures. Quaternions and their properties are defined at [Voi21]. They are 4 dimension vectors generated by the elements $\{1, i, j, k\}$ with the following multiplication rules

$$i^2 = a, j^2 = b, ij = -ji = k,$$

where the values chosen for a and b are -1 and $-p$, respectively.

It is also necessary some basic operations using quaternions. The conjugate of a quaternion $\alpha = w + xi + yj + zk$ is

$$\bar{\alpha} = w - xi - yj - zk,$$

the trace is

$$tr(\alpha) = tr(\bar{\alpha}) = \alpha + \bar{\alpha} = 2w,$$

and the reduced norm is

$$nrd(\alpha) = nrd(\bar{\alpha}) = \alpha\bar{\alpha} = w^2 + x^2 + p(y^2 + z^2).$$

The definitions of lattice, order, and ideal are all from [Cha+23]. A lattice is defined by a basis of linear independent quaternions, and elements in the lattice are represented using vectors in this basis, where the coefficients of these vectors are all integers. If the lattice is also a subring, it is an Order and, for an order O_a, if it does not exist O_b such that $O_a \subset O_b$, we call O_a a maximal order. From [Feo+22], an Eichler order is the intersection of two maximal orders.

Then, an ideal is a sublattice of an order. It is important to define the left-order of an ideal $O_L(I) = \{\alpha | \alpha I \subset I\}$ and equivalently the right-order of an ideal $O_R(I)$. The ideal is called the connecting ideal of O_L and O_R. An ideal has a norm equal to the greatest common divisor among its elements. From [Cha+23] we always can represent an ideal I as $I = O_L(I)\alpha + O_L(I)nrd(I)$ for some $\alpha \in O_L(I)$ which has the following notation $O\alpha + Onrd(I) = O\langle \alpha, nrd(I) \rangle$, this is an important aspect to be able to translate ideals into isogenies and will be mentioned again later on. We say that two ideals I and J are equivalent if there exists β such that $I = J\beta$.

2.9 Deuring correspondence

From the definition of supersingular elliptic curves [Sil09], we have that the endomorphism ring of an elliptic curve E is equivalent to a quaternion maximal order. For two elliptic curves E_1 and E_2 with endomorphism rings $End(E_1)$ and $End(E_2)$ and correspondent quaternion orders O_1 and O_2 there is also an equivalence between an isogeny $\phi : E_1 \to E_2$ of degree n and the ideal I with norm n and left order O_1 and right order O_2.

2.10 Computationally Hard Problems

The SQISign algorithm security is based in two mathematical problems. The first one is the endomorphism ring problem, the second is the isogeny path problem. As of today, both problems are considered computational difficult for both traditional and quantum computers, taking a long time to find a suitable answer using the known algorithms. [Wes22] proves that these problems are equivalent under polynomial reduction and also are equivalent to the problem of finding an ideal connecting two maximal Orders where the norm of the ideal is among the numbers of a set N.

2.10.1 Supersingular Isogeny Graph Path Problem

For a finite field $\mathbb{F}_{p^2}$ there is a limited number of different j-invariants where each of them represent an isomorphism class. Among the available j-invariants, $\lfloor \frac{p}{12} \rfloor + \epsilon$ where $\epsilon \in \{0, 1, 2\}$ represent supersingular elliptic curves isomorphism classes. Choosing a prime l, different from p, then, it is possible to construct a graph where, each vertex is a j-invariant representing a supersingular elliptic curve isomorphism class and each edge is an l-isogeny from one class to another. The resulting

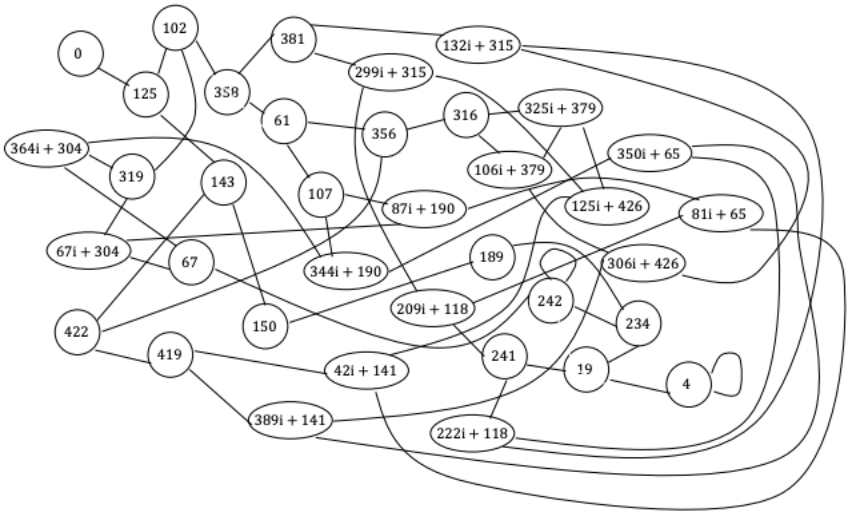

Figure 1: 2-isogeny graph for prime p=431, image from [Cos19]

graph is optimal expander [Wes22]. The problem is, given two vertexes in the graph, find a path between them. An image illustrating an isogeny graph is shown in Figure 1.

2.10.2 Endomorphism Ring Problem

The Endomorphism Ring Problem, as stated in [Wes22] is, given an elliptic curve E, find four endomorphisms that can generate the endomorphism ring $End(E)$ as a lattice.

2.10.3 Quaternion Equivalent Problems

The Endomorphism ring problem and the isogeny graph path problems have equivalent quaternion problems also discussed in [Wes22]. An expected polynomial solution to the quaternion equivalent of the isogeny path problem is known and is described at [Koh+14]. This algorithm is known as KLPT and some versions of it are used both in the key generation and signature steps of SQISign.

3 Algorithm Building Blocks

The SQISign algorithm uses the mathematical elements described in Section 2 in its modules, so it is necessary to be able to represent and perform operations on them. In this section, we discuss evaluating an isogeny on a curve point the algorithm for finding equivalent quaternions and how to translate between ideals and isogenies.

3.1 Computing Isogenies in SQISign

In many steps of the algorithm, it is important to be able to evaluate an isogeny on a point of an Elliptic Curve efficiently, as well as to obtain the isogeny codomain. In the algorithm specification [Cha+23] they have formulas to compute 2-isogenies and 4-isogenies.

In the case of odd-prime degree isogenies, it was chosen to use the $\sqrt{V}$élu's formula to evaluate the image of a point. However, the complexity of this algorithm increases significantly as the size of the prime also increases. As a result, it is only possible to use $\sqrt{V}$élu's formula on isogenies with

degree with sufficiently small prime factors (smooth). For an exposition about computing higher degree isogenies refer to [Ber+20].

3.1.1 Computing Quaternions in SQISign

The equivalences between supersingular elliptic curves and quaternion algebras described by the Deuring Correspondence allows performing part of the desired computations using quaternions. Specially, the KLPT algorithm [Koh+14] used both in the key generation and in verification, has as input an ideal I and allows obtaining an ideal J equivalent to I with different norms, this is done basically solving norm equations in some special situations. More specifically, there is the need to solve norm equations in p-extremal orders (defined in [Cha+23], here it is sufficient to know that the O_0 equivalent to E_0 is one of this p-extremal orders), in Eichler orders and in a left O-ideal, the last one can be performed combining the solutions of both the other situations using the Chinese Remainder Theorem [Cha+23]. There are two versions of KLPT used in SQISign, the first one, used in key generation is called KeyGenKLPT solves p-extremal order norm equations. The second version, used in the signature, is called SigningKLPT and solves norm equations of left O-ideals.

3.2 Converting Between Ideals and Isogenies

Here we are going to discuss how to obtain the equivalence between ideals and isogenies, starting from an ideal I connecting two orders O_1 and O_2. As stated in [Cha+23] an element $\alpha \in O_1$ corresponds to an endomorphism of the curve E_1 where $End(E_1)$ is the endomorphism ring of E_1, equivalent to O_1. It is possible then to evaluate α in the points of E_1. The set of points in E_1 that evaluates to ∞ under all elements of I is the kernel of ϕ_I the isogeny associated with I. Since we can represent I as $O_1\langle\alpha, nrd(I)\rangle$ we have the following equation that allows us to translate from an ideal I to $E[I]$ the kernel isogeny ϕ_I

$$E[I] = ker\alpha \cap E[nrd(I)].$$

There are two possible situations. The first one, I is a left-O_0 ideal. In this case, it is easier to find the equivalent isogeny. After finding an element of O_0 that generates this representation $O_0\langle\alpha, nrd(I)\rangle$, performing matrix operations using values that can be precomputed, it is possible to find a vector that gives the coefficients to calculate a generator of the kernel of $E[I]$ using the basis of $E[nrd(I)]$.

The second one, O_1 is an order different from O_0 and I has a large norm l^e where $e = fg$ and l is a prime number. In this case, the conversion is performed in a series of g steps. In each step, an ideal of norm l^f is used to calculate an isogeny of degree l^f using torsion groups basis and elements from the ideal left-order. The coefficient used to obtain the generator of the isogeny of the current step is used as part of the isogeny compression that is one of the algorithms output. The parameters l and f are implementation choices, and they have influence on the algorithms' performance.

In order to obtain the ideal equivalent to an isogeny we need a point P that generates the isogeny and two endomorphisms. Since there is interest only in translating isogenies starting from E_0 the values of the endomorphisms are precomputed and the translation can be done using a couple of matrix operations.

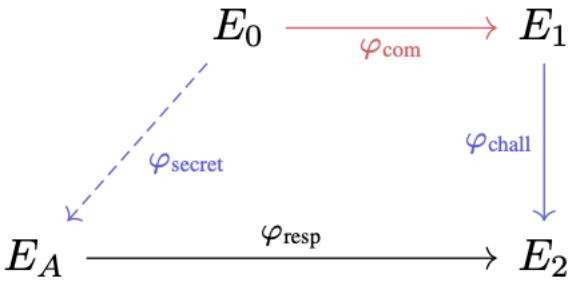

Figure 2: SQISign protocol, image from [Cha+23]

4 The SQISign algorithm

The SQISign algorithm can be divided in three modules. The first one allows generating a random pair of public and private key. The second one allows using the private key to generate a signature. The third one, given a message, a signature, and the correspondent public key, allows verifying the authenticity of the signature.

4.1 SQISign protocol

The SQISign algorithm aims to allow an agent here named Prover to interacts with another agent, the Verifier, and convince them about the Prover's identity revealing the least amount of information possible. In order to achieve this goal, first, the Prover randomly generates an isogeny ϕ from the curve E_0 to another curve E_P and publishes E_P as his public key. Then, he generates another isogeny ψ from E_0 to a random curve E_1. In this step called commitment, the Prover sends the Verifier the curve E_1. Receiving this information, the Verifier generates an isogeny θ from E_1 to another randomly generated curve E_2 and sends θ to the Prover as a challenge, the purpose of this step is to prevent the Prover from cheating. In the final step, the Prover uses all the isogenies to calculate $\omega = \hat{\phi} \circ \psi \circ \theta$ from E_P to E_2. The Verifier, finally, checks if ω is indeed an isogeny from E_P to E_2 and if $\hat{\theta} \circ \omega$ is cyclic and accepts the signature if so, or rejects otherwise. In order to remove the interaction between the Prover and the Verifier, it is possible to use the Fiat-Shamir transform [FS87] where the Prover uses a function with random/pseudo-random output to generate the challenge, here the function used is $H(msg, E_1)$ H being a hash function, E_1 being the codomain from the commitment isogeny and msg being the message to be signed. An image illustrating the SQISign protocol is shown in Figure 2.

4.2 Key Generation Module

The key generation algorithm is described at [Cha+23] Algorithm 25 and a simplified version is reproduced in Algorithm 1. As described in the algorithm, the steps necessary to generate the pair secret key public key is, first randomly choose a prime number and generate an ideal with norm equal to this prime number. Then, since this ideal cannot be used in the $\sqrt{V}$élu's formula, KeyGenKLPT is used to find a generator of an equivalent ideal that can be used. The generator is used to calculate the ideal, and the ideal is used to compute the secret isogeny and the image curve. Finally, using the secret isogeny it is possible to calculate a torsion basis in E_2 and a point Q with degree 2^f. The resulting public key is the image curve and the private key will be the generator, the torsion basis and the point Q.

Algorithm 1 SQISign.KeyGen(1^λ)

Input: 1^λ where λ is the security parameter
Output: Secret signing key sk and public verification key pk
Output: A boolean value indicating whether computation succeeded

Set found := False counter := 0
while found = False and counter < SQISIGN_keygen_attempts **do**
    counter := counter + 1
    Select a random KLPT_secret_key_prime_size-bit prime $D_{secret} \equiv 3 \pmod 4$
    Set I_{secret} to be a random ideal of norm D_{secret}
    Use KeyGenKLPT to find α, generator of $J_{secret} \simeq I_{secret}$
    if not found **then**
        Continue
    end if
    Use α to compute J_{secret}
    Use J_{secret} and pre-computed parameters to obtain the secret isogeny φ_{secret} with image curve E_A
end while
if found **then**
    Use φ_{secret} and pre-computed torsion basis information to compute torsion basis in E_A
    Use φ_{secret} to find a point Q with degree 2^f in E_A
    Use α, Q and the torsion basis in E_A as secret key sk
    Use E_A as public key pk
end if
return sk, pk, found

4.3 Signature Module

The signature algorithm is described at [Cha+23] Algorithm 26 and a simplified version is reproduced in Algorithm 2. It is implemented in the function *protocols_sign* which uses *protocols_commit* to generate an ideal correspondent to the commitment isogeny and uses *protocols_challenge* to generate an ideal correspondent to the challenge isogeny. Then, both ideals are used to generate the response. For the challenge isogeny and the response, the analysis will be about the Algorithm 26 [Cha+23].

The first step in *protocols_commit* is to randomly generate two numbers a and b, these numbers together with the values associated with computing endomorphism in E_0 are used to calculate a generator of the commitment isogeny I_{com}, and then the function *quat_lideal_make_primitive_then_create* is used to find I_{com}.

The values a and b and other precomputed values are then used to generate an isogeny, and the basis, which is part of the private key and was passed as a parameter, is evaluated in this isogeny. The resulting curve is used to normalize E_1 which is the codomain of the commitment isogeny. Both E_1 and I_{com} are then returned to *protocols_sign*.

The function *protocols_challenge*, as defined in the Algorithm 26 [Cha+23], starts generating the kernel of the challenge isogeny using the curve E_1 from the commitment stage and the message that is going to be signed.

Then, the kernel is used to generate the isogeny ϕ_{chall} and the image curve E_2. The kernel is used to generate its coefficients in the basis (P_1, Q_1) generated in the commitment. Subsequently, the coefficients are used to generate the ideal I_{chall}.

The next step is to generate a point P in E_1 such that $(Kernel, P)$ is a basis of the torsion subgroup $E[Degree_{challenge}]$ and get the image of $\phi_{chall}(P)$ through the challenge isogeny.

After, get a basis of the torsion subgroup $E_2[Degree_{challenge}]$. Then use the basis and $\phi_{chall}(P)$ to calculate b_1, b_2, s_1 and s_2.

Next, use $\phi_{chall}(P)$ to compute the dual isogeny $\hat{\phi_{chall}}$, use the values of b_1, b_2, s_1 and s_2 and $\phi_{chall}(P)$ to compute a basis of the torsion subgroup $E_2[Degree_{challenge}]$. Then get the image of the second point in the basis through the dual isogeny and use this image to calculate r.

In the response, the ideals $\bar{I_{secret}}$, I_{com} and I_{chall} are used to calculate K, an ideal J equivalent to K is calculated and if J has desired properties, J is converted to a compressed isogeny zip and the values b_1, b_2, s_1 and s_2, r and zip are returned as the signature.

4.4 Verification Module

The algorithm for the verification is described in the SQISign specification [Cha+23] Algorithm 27 and reproduced in Algorithm 3. The function responsible for doing the verification is *protocols_verif* which receives as parameters a signature, a public key, a message, and an unsigned integer. The function *protocols_verif* first calls *protocols_verif_unpack_chall* (Algorithm 28 [Cha+23]) and then calls *protocols_verif_from_chall* (Algorithm 29 [Cha+23]).

The function **protocols_verif_unpack_chall** deterministically generates a basis for the elliptic curve in the public key E_{pk} it receives as a parameter, using the function *ec_curve_to_basis_2*.

For NIST-I, signature zip has 14 10-byte integers, this values and the basis are used to calculate an isogeny chain. First start with a curve E, a basis $\langle P, Q\rangle$ for E and one of the integers i. Calculate a point of the kernel $K = p + i \times Q$ using the function *ec_ladder3pt*. Use the function

Algorithm 2 SQISign.Sign(sk, msg)

Input: Secret signing key sk and message msg
Output: Signature σ
Output: A boolean value indicating whether computation succeeded

Use sk to recompute secret isogenies I_{secret} and J_{secret}
Commitment.
Randomly generate a secret ideal I_{com}
Use I_{com} to generate the kernel of the commitment isogeny
Use the kernel to generate the commitment isogeny φ_{com} and image curve E_1
Use precomputed basis and φ_{com} to generate challenge basis
Challenge.
Use a hash function to generate K_{chall} the kernel of the challenge isogeny.
Use K_{chall} to generate φ_{chall} and the image curve E_2
Use K_{chall} to generate the corresponding ideal I_{chall}
Find a point P such that (K_{chall}, P) is a torsion basis in E_1
Compute $P' = \varphi_{chall}(P)$
Use P' to compute b_1, b_2 and the coefficients s_1 and s_2 that will help the Verifier compute $\varphi_{\hat{chall}}$
Find Q' such that $\langle P', Q' \rangle$ is a torsion basis in E_2
Compute $Q = \varphi_{\hat{chall}}(Q')$ and r such that $K_{chall} = [r]Q$
Response.
Compute $K = \bar{I_{secret}} \cdot I_{com} \cdot I_{chall}$ the response ideal
Find an ideal J equivalent to K such that $J \cdot \bar{I_{chall}}$ is cyclic
Compute the compressed isogeny zip corresponding to J
return (**zip**, r, (b_1, s_1, b_2, s_2))

ec_eval_even to compute the image curve of the isogeny and the image of the point P in the basis. Finally, use the function *ec_complete_basis_2* to generate a new basis for the image curve. This process is repeated for each one of the 14 integers and then, in the end, the resulting image curve is normalized generating a new curve E' through an isomorphism θ and the point $\theta(P)$ is calculated. The function then returns E' and $\theta(P)$.

The function *protocols_verif_from_chall* uses the image curve from the isogeny chain E' and the point P in E' generated by the *protocols_verif_unpack_chall* function to perform the verification. First it calculates the basis of two isogenies, for NIST-I one of the isogenies, ϕ_3, has degree 3^{36} and the other, ϕ_2, one has degree 2^{75} , both starting in E'. Then, using the basis, it calculates the kernel of both isogenies using *ec_ladder3pt*. Then, it tests if ϕ_2 is different from the curve received as a parameter. It evaluates ϕ_2, then evaluates an isogeny of degree 3^{36} from ϕ_2's image curve, the result is an isogeny of degree $2^{75} \times 3^{36}$. The next step is to calculate a basis for this resulting isogeny. Then the function calculates the challenge isogeny and its kernel K. Finally, calculate $Q \times r$ and check if it is equal to the challenge isogeny kernel, if it is, the signature is valid and if it is not, the signature is invalid.

Algorithm 3 SQISign.Verify(msg, σ, pk)

Input: A message msg, signature σ and pk the public key
Output: A boolean value indicating whether the verification passed

1: **zip**, r, s := σ
2: $E_2, Q_2 := \text{Decompress}_{\text{resp}}(\textbf{zip}, \text{pk})$
3: verified := $\text{DecompressAndCheck}_{\text{chall}}(\text{s}, E_2, Q_2, \text{r}, \text{msg})$
4: **return** verified

5 Implementation Aspects

5.1 Prime number choice

The SQISign algorithm has quite a few restrictions on the prime numbers that can be used. First, in order to be able to use the elliptic curve $E_0 : y^2 = x^3 + x$ and its correspondent quaternion order $O_0 = \mathbb{Z} \oplus i\mathbb{Z} \oplus \frac{i+j}{2}\mathbb{Z} \oplus \frac{1+k}{2}\mathbb{Z}$ in the algorithm, E_0 must be a supersingular elliptic curve. To guarantee this property, it is necessary that $p \equiv 3 \pmod 4$.

Then, in order to be able to use the $\sqrt{V}$élu's formula discussed in Section 3, as mentioned in the standard implementation [Cha+23], there is a slightly stricter restriction which imposes that $(p^2 - 1)$ has sufficiently many smooth prime factors, which means they have sufficiently small prime factors.

In order to be able to use 2 as the prime number in the ideal to isogeny translation, it is necessary to have a large power of 2 dividing $p^2 - 1$ [Cha+23]. All these restrictions makes it significantly hard to find suitable primes to use in SQISign and to scale it to higher degrees of security.

5.2 Failure Cases

Some routines used in the Key Generation and Signature phases of the SQISign protocol are non-deterministic, some of them can be made deterministic as mentioned in [Feo+20]. In special, Algorithms 8, 9, 10, 13, 14, 15, 16, 17, 19, 20, 25 and 26 described at [Cha+23] all return a boolean value indicating if the algorithm succeeded and in special, Algorithms 8, 10, 14, 15, 16, 17, 25 and

26 all have loops with variables limiting the maximum number of iterations, assuming a solution still has not been found. For the later case, the values limiting the number of iterations the loops can have, are choices made during the implementation and represent a tradeoff between success rate and time necessary to execute.

5.3 Constant Time

Functions where the execution time depend on the input can leak information about some secret data. In particular, in SQISign, multiplication of a point in an elliptic curve by a scalar is implemented using montgomery ladder. At each step, the use of masking helps keeping the time of execution independent of the points and scalars that are being multiplied.

6 Standard Implementation

An implementation of the SQISign algorithm using the C programming language was submitted to NIST together with the algorithm specification and is available in [Cha+23]. In this section, some details about this implementation will be discussed. The NIST-I level of security parameters will be used when convenient to illustrate some details of the implementation.

6.1 Parameters used in the algorithm

In Section 4.B.3 of its call for additional digital signature schemes [1], NIST defines five levels of security for the algorithms which would be proposed. SQISign [Cha+23] implements levels 1, 3 and 5. The parameters used can be derived from the prime number p over which the elliptic curves used in the algorithm will be defined and B the bound on the prime factors of T, which represents the accessible torsion and is used for choosing some torsion basis and consequently isogenies that are used in the algorithm. The values of p, in hexadecimal, and B for each level of security used in the standard implementation are reproduced below

NIST I:

$$p = \texttt{0x34e29e286b95d98c33a6a86587407437252c9e49355147ffffffffffffffff}$$
$$B = 2000$$

NIST III:

$$p = \texttt{0x3df6eeeab0871a2c6ae604a45d10ad665bc2e0a90aeb751c722f669356ea4684c}$$
$$\texttt{6174c1ffffffffffffffffffffff}$$
$$B = 48000$$

NIST V:

$$p = \texttt{0x255946a8869bc68c15b0036936e79202bdbe6326507d01fe3ac5904a0dea65faf0a}$$
$$\texttt{29a781974ce994c68ada6e1fffffffffffffffffffffffffffffffffffff}$$
$$B = 320000$$

[1] The NIST additional call for digital signature schemes is available at https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf

6.2 Performance and Sizes

Regarding performance of the SQISign standard implementation, there is a benchmark script in the files submitted to NIST which can be used to measure performance. In the algorithm specification [Cha+23] they also performed some tests, the results are reproduced in Table 1.

Parameter set	KeyGen	Sign	Verify
Reference implementation (with default GMP installation)			
NIST-I	2′834	4′781	103
NIST-III	21′359	38′884	687
NIST-V	84′944	160′458	2’051
Reference implementation (with GMP –disable-assembly)			
NIST-I	3′728	5′779	108
NIST-III	23′734	43′760	654
NIST-V	91′049	158′544	2′177
Assembly-optimized implementation for Intel Broadwell or later			
NIST-I	1′661	2′370	37

Table 1: Performance achieved using the SQISign standard implementation, the unity of the values shown is 10^6 CPU cycles on an Intel Xeon Gold 6338 CPU (Ice Lake), compiled on Ubuntu with clang version 14. The table is reproduced from [Cha+23]

The key and signature sizes for levels of security I, III and V are also mentioned in [Cha+23] and reproduced in Table 2.

7 Other Implementations

Implementations beyond the standard one were really hard to find. Being more specific, it was found a paper with a C implementation using NIST I level of security [Lin+23] and a Sagemath implementation with more educational purpose[2]. Since the performance of the Sagemath implementation is expected to be much worse than the standard one, it is not going to be mentioned. The performance for [Lin+23] is shown in Table 3.

8 SQISign Based Algorithms

Besides SQISign, some other algorithms were proposed based on the same problems, but with different purposes. Among them there was Supersingular isogeny Diffie–Hellman key exchange (SIDH or SIKE) which is an algorithm that allows two parties to agree on a secret piece of information. For a friendly introduction to how SIKE works, refer to [Cos19].

[2]The Sagemath implementation is available at https://learningtosqi.github.io/

Parameter set	Public key	Secret key	Signature
NIST-I	64	782	177
NIST-III	96	1138	263
NIST-V	128	1509	335

Table 2: Key sizes for NIST I, III and V level of security. Table reproduced from [Cha+23]

Phase	Previous work	This work			
		without precomp.	Speedup	with precomp.	Speedup
Keygen	1491.4	1409.8	5.47%	1525.3	-2.27%
Sign	2371.4	2162.7	8.80%	1944.0	18.02%
Verify	36.7	27.4	25.34%	27.4	25.34%

Table 3: Performance achieved using the SQISign implementation of [Lin+23]. The unity of the values shown is 10^6 CPU cycles on Intel(R) Core(TM) i9-12900K 3.20 GHz with TurboBoost and hyperthreading features disabled. The table is reproduced from [Lin+23]

For the context of SIKE, however, the information exchanged between the two agents during the protocol, together with higher extension fields, which allow representing efficiently arbitrary isogenies could speed up the process of recovering the key. As a consequence, an attack such as described at [CD22] was able to completely break SIKE and recover the secret information in feasible time with everyday use computers.

The idea was later further developed and allowed the creation of new algorithms based on SQISign using higher extension field mathematics. One of the new algorithms based in SQISign is called SQISignHD [Dar+23] it has two versions, one using four dimension isogenies, which is faster, and the other using eight dimension isogenies, which is more secure. Even though, as mentioned by the authors of SQISignHD, as the dimension of the isogenies increase, computing them also becomes exponentially costly, an algorithm using two dimension isogenies was not yet proved secure. Recently, two algorithms SQISign2D-West [Bas+24] and SQISign2D-East [NO24] were simultaneously proposed by different authors, both using two dimension isogenies. Since both algorithms have similar properties, we will focus on one of them, SQISign2D-West.

In this section we discuss some differences between the SQISign specification submitted to NIST in the Post-Quantum Cryptography Standardization [Cha+23], SQISignHD [Dar+23] and SQISign2D-West [Bas+24]. For SQISignHD, as in the proposing paper, we say FastSQISignHD to refer to the 4 dimension isogenies version.

8.1 Prime number choice

The SQISignHD algorithm also uses the elliptic curve E_0 in its implementation, so it is necessary to have $p \equiv 3 \pmod 4$.

As mentioned in [Dar+23], the usage of higher dimension isogenies allows describing an isogeny by listing only the image of a few well-chosen points. This improvement over SQISign allow the choice of primes of the form $p = cl^f l'^{f'} - 1$ with $l \neq l'$ two primes, $c \in \mathbb{N}^*$ small. In practice, $l = 2$ and $l' = 3$. An example of prime used in FastSQISignHD for NIST-I is $p = 13 \times 2^{126} \times 3^{78} - 1$.

Among the other algorithms discussed in this section, SQISign2D-West has the most relaxed restrictions on primes to be chosen. As SQISignHD [Dar+23] and SQISign [Cha+23] it also needs that $p \equiv 3 \pmod 4$ again, to make sure that the elliptic curve E_0 is supersingular. This time, however, the prime only has to be of the form $p = c \times 2^e - 1$ where c is a small cofactor, scaling really well for higher security levels.

8.2 Output Sizes and Execution Times

As mentioned in [Dar+23], for the NIST-I level of security, the size of the signature generated by the algorithm FastSQISignHD is 109 bytes, versus 177 bytes from the standard implementation of SQISign is a significant improvement.

Regarding execution time, in the FastSQISignHD it was chosen to use the chain of isogenies $\varphi \circ \tau \circ \hat{\psi} : E_1 \to E_2$ in the signing phase, where the isogenies represent the challenge isogeny, the secret isogeny and the dual of the commitment isogeny, respectively. Then, during the verification phase, it is necessary to compute the 4 dimension isogeny equivalent to this chain. As a result, the verification time is significantly longer than the signing time. The implementation for the NIST-I level of security in [Dar+23] achieved in average $28ms$ on an Intel(R) Core(TM) i5-1335U 4600MHz CPU using C-language for the signing phase and around $600ms$ using SageMath for the verification phase. There is space for improvement in both processes.

As mentioned in [Bas+24], for the NIST-I level of security, the size of the signature generated by the algorithm is 148 bytes, a little longer than the SQISignHD signatures, but still shorter than the SQISign 177 bytes ones. It was still mentioned by the authors that efficiency was prioritized over signature size.

Regarding execution time, the authors of [Bas+24] ran benchmarks on an Intel Xeon Gold 6338 (Ice Lake) CPU clocked at 2 GHz with turbo boost disabled. For the stricter security proof version we have, in 10^6 clock cycles, 290 for the signature in NIST-I and 25 for the verification, also in NIST-I. Both modules show a very relevant increase in performance versus SQISign. The signing is still slower than the SQISignHD signature, and for the verification it was chosen not to compare since SQISignHD only has the SageMath implementation of the verification. For the heuristic security version of SQISign2D-West, the numbers shown in [Bas+24] are even better, surpassing the other two algorithms in every module.

8.3 Security Assumptions

The algorithms mentioned in this section all rely on the hardness of the endomorphism ring problem, which is equivalent under polynomial reduction to the path-finding problem in the l-isogeny graph. As mentioned in [Cha+23] the best known solutions to these problems have time complexity proportional to $\tilde{O}(\sqrt{p})$ in classical computers and proportional to $\tilde{O}(p^{\frac{1}{4}})$ in quantum computers, where p is the prime number used in the algorithm.

There are some additional security features that are desirable for signature schemes, this features will be discussed below. They might differ from one algorithm to another. We use the definition of witness w, the secret information that is known only by the respective Prover. We call a statement x some public information that is associated with a witness.For the algorithms in this section, the witness is the secret key and the statement is the public key. Finally, a transcript is a tuple with the commitment isogeny, the challenge isogeny and the response isogeny.

8.3.1 Correctness

The correctness property states that a Prover with a valid proof and behaving honestly always has his proof accepted by the Verifier, this property is aimed by all the algorithms.

8.3.2 Special Soundness

We then say that a signature scheme has special soundness if someone who only knows a statement x cannot generate a valid proof for it.

8.3.3 Honest-verifier zero-knowledge

Maybe the biggest difference between the security assumptions of the algorithms discussed in this section is the honest-verifier zero-knowledge, which uses a different type of oracle proof in each of

the algorithms.

In the SQISign specification [Cha+23] it is assumed that an adversary A has access to a random transcript generation oracle (equivalent to eavesdrop communications between the Prover and the Verifier), and then uses the transcripts generated by this oracle to try to generate valid proofs without the knowledge of the secret information (the witness w).

In the SQISignHD algorithm [Dar+23] the adversary is given access to a random uniform good degree isogeny oracle (RUGDIO) which, given a supersingular elliptic curve E can generate an efficient representation of a l^e-good degree prime to l' isogeny from E to another random elliptic curve E'. An adversary with access to this oracle is believed not able to generate valid proofs without knowing the Prover's witness w.

In the SQISign2d-West algorithm [Bas+24] the property is proven in two ways, the first one giving the adversary access to a uniform target oracle (UTO) and the second one giving them access to a fixed degree isogeny oracle (FIDIO). The first one takes as input an elliptic curve E and an integer $N \geq \frac{2\sqrt{2p}}{\pi}$ and outputs an efficient representation of a random isogeny to another random curve E'. This isogeny has degree $d \leq N$. The second one also takes as input an elliptic curve E and an integer N and outputs an efficient representation of a random isogeny to another random curve E'. This isogeny has degree $d = N$.

9 Conclusion

Signature schemes are relevant to a wild range of applications, and the Shor's algorithm [Sho99] will be able to break the currently most used schemes based on factoring or computing discrete logarithms as soon as a large-scale quantum computer is available. In order to keep providing security to the users, without compromising usability, it is necessary to explore new ways to implement signature schemes that could be resistant to quantum computers. These solutions cannot rely on the hardness of factoring large numbers and equivalent problems and, instead, have to find new cryptographic premises. In this context, the problem of finding a l-path in an isogeny graph is still considered a computationally hard problem for quantum computers to solve, but algorithms using isogenies in the cryptography context lack the same amount of expertise of algorithms using elliptic curves, as an example. Among the candidates in the NIST Post-Quantum Standardization Contest, SQISign is the only one which uses isogenies between elliptic curves as its cryptographic premise. SQISign, being the only one of its kind, has a lot of potential, specially for devices with limited resources, since it has very short key and signature sizes, but, as it is currently implemented, the signature times are still long, and the primes that can be used are restricted which makes it not scale well to higher levels of security. The new variants created after [CD22] show that maybe using higher dimension isogenies is a way to achieve performance improvements and overcome the prime number restrictions of the original SQISign proposal, however these variants still need time to have their security and functioning better understood. Maybe there are other ways to make isogeny algorithms more viable, such as exploring hardware specific implementations, exploring parallelism. This could lead to lower execution times, even smaller keys and different security proofs. For now, we have to wait and hope that the future of isogenies is as bright as it seems.

Acknowledgement

The author thanks the Cryptography Research Centre at the Technology Innovation Institute for financial support during the development of this work.

References

[FS87] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology — CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194. ISBN: 978-3-540-47721-1.

[IHH91] Niven I., Zuckerman H., and Montgomery H. *An Introduction to the Theory of Numbers*. https://books.google.com.br/books?id=V52HIcKguJ4C: Wiley, 1991.

[Sho99] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Review* 41.2 (1999), pp. 303–332. DOI: 10.1137/S0036144598347011. eprint: https://doi.org/10.1137/S0036144598347011. URL: https://doi.org/10.1137/S0036144598347011.

[Sil09] J.H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer New York, 2009. ISBN: 9780387094946. URL: https://books.google.com.br/books?id=Z90CA_EUCCkC.

[Koh+14] David Kohel et al. *On the quaternion ℓ-isogeny path problem*. Cryptology ePrint Archive, Paper 2014/505. https://eprint.iacr.org/2014/505. 2014. URL: https://eprint.iacr.org/2014/505.

[CS17] Craig Costello and Benjamin Smith. *Montgomery curves and their arithmetic: The case of large characteristic fields*. Cryptology ePrint Archive, Paper 2017/212. https://eprint.iacr.org/2017/212. 2017. URL: https://eprint.iacr.org/2017/212.

[Cos19] Craig Costello. *Supersingular isogeny key exchange for beginners*. Cryptology ePrint Archive, Paper 2019/1321. https://eprint.iacr.org/2019/1321. 2019. URL: https://eprint.iacr.org/2019/1321.

[Ber+20] Daniel J. Bernstein et al. *Faster computation of isogenies of large prime degree*. Cryptology ePrint Archive, Paper 2020/341. https://eprint.iacr.org/2020/341. 2020. URL: https://eprint.iacr.org/2020/341.

[Feo+20] Luca De Feo et al. *SQISign: compact post-quantum signatures from quaternions and isogenies*. Cryptology ePrint Archive, Paper 2020/1240. https://eprint.iacr.org/2020/1240. 2020. URL: https://eprint.iacr.org/2020/1240.

[Voi21] J. Voight. *Quaternion Algebras*. Graduate Texts in Mathematics. Springer International Publishing, 2021. ISBN: 9783030566944. URL: https://books.google.com.br/books?id=Gro1EAAAQBAJ.

[CD22] Wouter Castryck and Thomas Decru. *An efficient key recovery attack on SIDH*. Cryptology ePrint Archive, Paper 2022/975. https://eprint.iacr.org/2022/975. 2022. URL: https://eprint.iacr.org/2022/975.

[Feo+22] Luca De Feo et al. *New algorithms for the Deuring correspondence: Towards practical and secure SQISign signatures*. Cryptology ePrint Archive, Paper 2022/234. https://eprint.iacr.org/2022/234. 2022. URL: https://eprint.iacr.org/2022/234.

[Wes22] Benjamin Wesolowski. "The supersingular isogeny path and endomorphism ring problems are equivalent". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 1100–1111. DOI: 10.1109/FOCS52979.2021.00109.

[Cha+23] Jorge Chavez-Saab et al. *SQIsign*. Technical report, National Institute of Standards and Technology. Available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures`. 2023.

[Dar+23] Pierrick Dartois et al. *SQISignHD: New Dimensions in Cryptography*. Cryptology ePrint Archive, Paper 2023/436. `https://eprint.iacr.org/2023/436`. 2023. URL: `https://eprint.iacr.org/2023/436`.

[Lin+23] Kaizhan Lin et al. *A Faster Software Implementation of SQISign*. Cryptology ePrint Archive, Paper 2023/753. `https://eprint.iacr.org/2023/753`. 2023. URL: `https://eprint.iacr.org/2023/753`.

[Bas+24] Andrea Basso et al. *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`.

[NO24] 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`.