Towards a constant-time implementation of isogeny-based signature, SQISign

David Jacquemin[1], Anisha Mukherjee[1], Péter Kutas[2], and Sujoy Sinha Roy[1]

[1]IAIK, Graz University of Technology, Austria
[2]Eötvös Loránd University and University of Birmingham

Abstract. Isogeny-based cryptographic constructions are well-known in the domain of post-quantum security. One such instance is SQISign, that boasts the most compact key and signature sizes among all post-quantum signature schemes. However, its current implementation is not free from side-channel vulnerabilities. At certain steps within the signing procedure, it relies on Cornacchia's algorithm to represent an integer as a sum of squares of two integers. This algorithm in turn uses a 'half-GCD' subroutine that is based on a non-constant time version of the Euclidean algorithm. We show that if inputs of Cornacchia's algorithm leaks, then one can retrieve the signing key in polynomial time. We propose two timing attack-resistant versions of Cornacchia's algorithm. The first version is based on a lattice reduction algorithm. We show that randomising the starting basis with a unimodular matrix would make the execution time independent of the input. The second version uses a constant-time 'half-GCD' algorithm that runs a fixed number of times for a given upper bound on the size of inputs.

Keywords: isogeny-based cryptography · SQISign, side-channel analysis · isogeny signature · constant-time implementation

1 Introduction

The currently deployed cryptographic primitives largely rely on the mathematical hard problems of integer factorisation and discrete logarithms. Peter Shor in 1997 published a breakthrough algorithm [31] to factor large integers (which was super-polynomial in time in classical computers) in polynomial time with the use of quantum computing. Large-scale quantum computers therefore threaten to render these primitives obsolete. Hence, in 2016 NIST started the procedure to standardize Key Encapsulation Mechanisms (KEMs) and digital signature schemes [26]. The process is currently in Round 4 and the first winners have been announced in 2022. Three digital signature schemes were chosen for standardization: Dilithium [16], Falcon and SPHINCS+ [5]. While all these schemes have their merits, NIST deemed the portfolio of signature schemes not diverse enough and announced a new call [27] for 2023, exclusively devoted to post-quantum signature schemes.

Isogeny-based cryptographic primitives have become popular in post-quantum cryptography. Isogeny-based schemes date back to Couveignes's seminal paper

[11] which was then rediscovered by Rostovtsev and Stolbunov [30]. Improving on Couveignes's original approach, De Feo and Galbraith constructed SeaSign [13] using the Fiat-Shamir protocol with aborts technique. Then Beullens, Kleinjung and Vercauteren proposed CSi-FiSh [7] which uses Couveignes's original idea combined with a record class group computation. CSi-FiSh achieves respectable performance but it is currently infeasible to instantiate it for higher security levels and it potentially does not reach NIST level I security [28],[8]. An alternative option came in the form of SCALLOP [18] but it is also currently too slow for applications. All these signatures are based on cryptographic group actions (of which CSIDH [10] and SCALLOP are examples [18]).

Taking inspiration from SIDH one can also build signature schemes based on the pure isogeny problem which could actually remain unaffected by recent SIDH attacks [9],[25],[29], such as the work by [12]. This scheme however is based on an identification system with low soundness and thus, has to be repeated many number of times resulting in big signature sizes.

A completely different approach for constructing signatures from isogenies was proposed by Galbraith, Petit and Silva [20](GPS). Their signature relies on the observation that one can compute isogenies between curves of known endomorphism rings. Based on this observation they designed an identification scheme reminiscent of graph isomorphism. For this they rely on a quaternion version of the isogeny path-finding problem which was efficiently solved by Kohel, Lauter, Petit and Tignol hence dubbed the KLPT algorithm [22]. The advantage of the GPS signature scheme is that its security is based on the endomorphism ring problem which is the fundamental hard problem in isogeny-based cryptography. On the other hand, it has a constrained challenge space as it allows bit-long challenges.

In 2020 De Feo, Kohel, Leroux, Petit and Wesolowski proposed SQISign [14] which improves upon the GPS scheme in many ways and provides a much better performance. SQISign is a particularly promising candidate due to its low bandwidth requirements as it is the most compact post-quantum signature to date. Being a fairly recent addition to the list of isogeny signatures, the scheme is yet to undergo rigorous cryptanalysis. It also lacks a side-channel resistant implementation which is important for practical applications.

1.1 Our contributions

The current implementation of SQISign is not constant-time makes it vulnerable to side-channel attacks. Our main contributions are not just limited to identifying one such vulnerability; we also propose suitable alternatives to shield the implementation against such weaknesses.

We highlight an instance of the non-constant time Cornacchia's algorithm within SQISign's signing routine that could make it susceptible to fatal leakages. As the first contribution, we present a detailed polynomial-time method for an adversary to recover the secret signing key of SQISign if she is able to exploit these 'leaky' algorithms. We show that the knowledge of Cornacchia's inputs

allows the attacker to obtain a feasibly 'small' set of possible intermediate values, which will ultimately reveal the entire signing key.

Our next contribution is two-fold. As a way of mitigating the proposed vulnerability, we recommend two timing attack-resistant alternatives to the Cornacchia's algorithm: a randomised lattice reduction algorithm, and, a constant-time half-GCD algorithm. We argue that solving an equation of the form '$x^2+y^2 = m$' using Cornacchia's algorithm is equivalent to finding the shortest vector in a suitable two-dimensional lattice. We add the basis randomisation step to ensure that the internal executions no longer correlate with the initial secret-dependent inputs of the algorithm. Finally, we propose a constant-time half-GCD algorithm that could be used in place of the current non-constant time version. We show that introducing an extra parameter to control the execution steps and making small tweaks to the underlying idea of the usual Euclid's algorithm are actually enough to give shape to our proposed algorithm.

1.2 Organisation of the paper

In Sec. 2, we give all the important background information necessary for our paper. The design rationale of SQISign spans over a vast range of topics related to elliptic curves and quaternion algebra. Many minute details of its construction are not intuitive and can only be understood only after an in-depth study of the scheme. We provide explanations only for a handful of carefully selected topics to make our results comprehensible to the reader. Sec. 3 discusses a weakness of the current SQISign implementation and presents a way to retrieve the signing key using side-channel leakages. We dedicate Sec. 4 to suggest alternative timing-attack resistant algorithms that could be used to replace Cornacchia's algorithm. We provide performance results in Sec. 5. In Sec. 6 we provide insights into future directions that would motivate greater research towards exploring the potentials of SQISign and realising a more practical, side-channel resistant implementation.

2 Preliminaries

In this section we cover certain mathematical and algorithmic topics necessary to present our results. For more details on elliptic curves we urge the reader to refer to [32] and on quaternion algebras we refer to [35].

2.1 Supersingular elliptic curves

Let E_1 and E_2 be elliptic curves defined over some finite field $\mathbb{F}_q$. An isogeny is a non-zero rational map that sends the point of infinity of E_1 to the point of infinity of E_2. Alternatively, one can also define isogenies as rational maps which are simultaneously group homomorphisms. An isogeny is a finite map of curves thus induces a finite field extension on the function fields. An isogeny is called separable if the said field extension is separable. The degree of the isogeny is defined as the degree of the field extension. If an isogeny is separable, then its

degree is equal to the size of its kernel. For every isogeny $\phi : E_1 \to E_2$ there exists a dual isogeny $\hat{\phi} : E_2 \to E_1$ which has the same degree as ϕ and $\phi \circ \hat{\phi}$ is the multiplication-by-deg(ϕ) map(this is also true if they are composed in the other order). Two elliptic curves E_1 and E_2 are isomorphic if and only if there exists a degree 1 isogeny between them. To every elliptic curve E defined over $\mathbb{F}_q$ one can associate an element of $\mathbb{F}_q$ called the j-invariant, denoted by $j(E)$. Two elliptic curves defined over $\mathbb{F}_q$ are isomorphic if and only if their j-invariants coincide.

An endomorphism of E is an isogeny from E to itself. In order to obtain a ring structure, the zero map is also considered to be an endomorphism. Endomorphisms thus form a ring under addition and composition. If E is defined over $\mathbb{F}_q$, then its endomorphism ring is never equal to $\mathbb{Z}$ (i.e., only contains scalar multiplications) but in most cases it is still commutative (an order in a quadratic field). If the endomorphism ring of E is commutative, then we call E *ordinary*, otherwise E is *supersingular*. In this paper we will only be looking at supersingular elliptic curves. Supersingular elliptic curves can all be defined over $\mathbb{F}_p^2$. One can detect supersingularity efficiently by computing its number of points over $\mathbb{F}_{p^2}$ (as an equivalent definition of supersingularity is that the trace of Frobenius is divisible by p).

2.2 Quaternion algebras and the Deuring correspondence

As mentioned before, the endomorphism ring of a supersingular elliptic curve is a non-commutative ring. In this subsection, we give more detailed about the endomorphism ring and introduce an important categorical equivalence, called the Deuring correspondence. A rational quaternion algebra H is a central simple algebra of dimension four over $\mathbb{Q}$. It always has a presentation $i^2 = a, j^2 = b, ij = -ij$ where $1, i, j, ij$ constitute a $\mathbb{Q}$-basis and $a, b \in \mathbb{Q}^*$: $H(a, b) = \mathbb{Q}+i\mathbb{Q}+j\mathbb{Q}+k\mathbb{Q}$. An order in a quaternion algebra is a subring that contains 1 and also contains a Q-basis of the algebra. An order is called maximal if it is maximal with respect to inclusion.

Let $B_{p,\infty}$ be the quaternion algebra ramified at p and infinity. In the special case where $p \equiv 3 \pmod 4$ this just amounts to $i^2 = -1$ and $j^2 = -p$. The endomorphism ring of a supersingular curve over $\mathbb{F}_p^2$ is a maximal order in $B_{p,\infty}$. Furthermore, every maximal order in $B_{p,\infty}$ is the endomorphism ring of a supersingular elliptic curve. It is also well-understood when non-isomorphic curves have isomorphic endomorphism rings, namely if and only if they are Frobenius conjugates. The Deuring correspondence is a correspondence between isomorphism classes of supersingular elliptic curves over $\mathbb{F}_{p^2}$ and isomorphism classes of maximal orders in $B_{p,\infty}$. The Deuring correspondence is two-to-one in most cases and one-to-one if and only if the curve can be defined over $\mathbb{F}_p$. This relation has considerably more structure than just some map between sets. One can consider supersingular elliptic curves together with isogenies, so we need to define the quaternion analogue of an elliptic curve isogeny.

The analogue of an isogeny with domain E is a left ideal of End(E). The way to associate a left ideal to an isogeny is as follows. Let $\phi : E \to E_A$ be an isogeny whose kernel is a subgroup G. Then one can take the set of all endomorphisms of

E which vanish on G. It is easy to see that this set is a left ideal. In the reverse direction we can take a left ideal I of End(E) and intersect all the kernels of the endomorphisms in I. This provides us a subgroup which we denote by $E[I]$. Then there is a unique isogeny with kernel $E[I]$ which will be the isogeny associated to I. The norm of the ideal is defined as the greatest common divisors of all the norms in I and is equal to the degree of the associated isogeny.

A natural question is how do we get the "codomain" for a given left ideal I. This is provided by the right order $\mathcal{O}_r(I) = \{\beta \in B_{p,\infty} | I\beta \subseteq I\}$. A connecting ideal between $\mathcal{O}_1$ and $\mathcal{O}_2$ is a left ideal of $\mathcal{O}_1$ whose right order is isomorphic to $\mathcal{O}_2$. An Eichler order is the intersection of two maximal orders. It encodes a particular connecting ideal as $\mathcal{O}_1 \cap \mathcal{O}_2 = \mathbb{Z} + I$ where I is a left ideal of $\mathcal{O}_1$ and a right ideal of $\mathcal{O}_2$. $\mathcal{O}_1 \cap \mathcal{O}_2$ is a $\mathbb{Z}$ is a sub-lattice of $\mathcal{O}_1$ and $\mathcal{O}_2$ of the same index (sometimes called the level) which is equal to the norm of the ideal I.

2.3 KLPT and SQISign

In isogeny-based cryptography the natural algorithmic problem is finding isogenies between supersingular elliptic curves. A more specific problem is when the degree, say, d of the isogeny is also specified. When d is small this problem is quite easy, hence, the interesting case is when d is large. In that case however, just writing down the output (essentially polynomials of degree d) might be problematic for a generic d. Therefore, it is rather natural to restrict to degrees which are smooth and quite naturally to the case where $d = l^k$ and l is small prime (e.g., 2 or 3).

These problems are all thought be hard even for a quantum computer. Due to Deuring correspondence, there is a natural quaternion analogue of this problem. Namely given two maximal orders, find a connecting ideal of norm l^e. Such a connecting ideal should exist for a large enough e. Even though it is the exact analogue of a hard problem, it admits a polynomial time algorithm, discovered by Kohel, Lauter, Petit and Tignol [22]. We briefly sketch the idea of the KLPT algorithm here.

For simplicity we will assume that $p \equiv 3 \pmod 4$ (the general case is not much harder). The first observation is that if one wants to find a path between two maximal orders, then it is actually enough to connect both maximal orders to one specific order and concatenate the paths. This special order, denoted by $\mathcal{O}_0$ is the endomorphism ring of the elliptic curve $y^2 = x^3 + x$ and contains $1, i, j$ (it is actually generated as a $\mathbb{Z}$-module by $1, i, (1 + j)/2$ and $(i + ij)/2$). From now on we assume that we are looking for a connecting ideal between $\mathcal{O}_0$ and some other maximal order $\mathcal{O}$. One finds a connecting ideal I_0 using an algorithm of Kirschmer and Voight [21]. Naturally, the norm of this ideal is not likely to be of the form l^e, hence our goal is to find an equivalent left ideal of $\mathcal{O}_0$ such that its norm is l^e. This is basically equivalent to finding some element in I_0 whose norm is $n(I_0) = l^e$. The next important intermediate step is that one finds an equivalent ideal of prime norm N. This is easy as one just scans through elements of I_0 and since primes are relatively dense, one will find a suitable element very efficiently. So our goal is to find $\beta \in I$ such that

$n(\beta) = Nl^e$. The next observation is that $\mathcal{O}_0/N\mathcal{O}_0$ is isomorphic to $M_2(\mathbb{Z}/N\mathbb{Z})$ and an explicit isomorphism can be computed efficiently. Take into consideration an endomorphism $\gamma \in \mathcal{O}_0$ of norm Nl^{e_0} for some e_0 (this involves solving a very simple quadratic equation). Then one can view $\mathcal{O}_0\gamma$ as a left ideal in $\mathcal{O}_0/N\mathcal{O}_0$ which is going to be a proper left ideal. I can also be viewed as left ideal in $\mathcal{O}_0/N\mathcal{O}_0$ and is also a proper left ideal as $n(I) = N$. In $M_2(\mathbb{Z}/N\mathbb{Z})$ every proper left ideal is isomorphic and differs only by a right multiplication which can be computed easily. Thus one can compute $\mu \in \mathcal{O}_0$ such that $(\mathcal{O}_0\gamma)\mu \equiv I \pmod{N\mathcal{O}_0}$. Here everything is only determined modulo $N\mathcal{O}_0$, hence, there is large variety of choices for μ. If one manages to choose a μ whose norm is l^{e_1}, then $\beta = \gamma \cdot \mu$ is going to be an appropriate choice. However, in general, this lifting problem can be hard. The observation of KLPT is that the lifting problem is actually easy if μ is a $\mathbb{Z}$-linear combination of j and ij (we will call this set $j\mathbb{Z}[i]$). This algorithm is called StrongApproximation in [22]. Luckily μ can be chosen that way. We provide a simple reasoning in the case where N is a prime congruent to 3 mod 4. There are $N + 1$ left ideals in $M_2(\mathbb{Z}/N\mathbb{Z})$ and, in $j\mathbb{Z}[i]$ there are (modulo $N\mathcal{O}_0$) $N^2 - 1$ invertible elements. Since only $N - 1$ stabilize the left ideal, this action should be transitive on the set of left ideals. The crucial detail of the KLPT algorithm is the lifting procedure which only works in this specific scenario, as one would need $j\mathbb{Z}[i]$ to be contained in $\mathcal{O}_0$.

One way to interpret KLPT is that if one has two supersingular elliptic curves with known endomorphism rings, then one can compute an isogeny between them (a task that is deemed hard otherwise). This motivates the following sigma protocol (we provide a high level description) which is the high-level idea of SQISign [14]. The public key is a supersingular elliptic curve E and let E_0 be the supersingular elliptic curve, $y^2 = x^3 + x$. Let $\mathrm{End}(E_0) = \mathcal{O}_0$ and let $\mathrm{End}(E) = \mathcal{O}$. Here $\mathcal{O}_0$ and $\mathcal{O}$ are connected by a secret left ideal I_τ of large prime norm. The secret key is the endomorphism ring of E. The main steps of the protocol are the following:

1. Prover computes an isogeny, $\phi : E_0 \to E_1$ and sends E_1 to the verifier.
2. Verifier sends a challenge isogeny, $\psi : E_1 \to E_2$ to the verifier.
3. Prover responds with an isogeny between E and E_2 of the degree l^e. Verifier accepts if the isogeny is indeed between those two curves and has the correct degree (of the form l^e).

The degree condition is necessary as otherwise a malicious prover could take an isogeny from E to E_1 as a commitment and then concatenate it with the challenge isogeny. Since one can translate endomorphism rings through isogenies [20], the endomorphism ring of E_2 can be computed by the prover. The naive approach here is to use the KLPT algorithm to provide an isogeny between E and E_2. However, the KLPT isogeny has the peculiar feature that it always goes through E_0. This could reveal the secret key (i.e., the endomorphism ring of E) after one response. SQISign uses a modified version of KLPT which connects E and E_2 in a direct fashion, not leaking (conjecturally) any information about the endomorphism ring of E. This can be done by deriving a quaternion analogue of commutative isogeny diagrams. One constructs a left $\mathcal{O}_0$-ideal of the correct

norm and pushes it forward via I_τ to a left ideal of $\mathcal{O}$. Of course the difficulty here is to ensure that the right order of this left ideal is isomorphic to $\mathrm{End}(E_2)$. This is ensured by using the Eichler order $\mathcal{O}_0 \cap \mathcal{O}$ and other techniques developed in KLPT.

In order to be able to respond to the verifier one has to translate the aforementioned ideal into an isogeny. There are two issues that arise in practice. First, the norm of the ideal obtained this way is large, hence its kernel will be defined over a large extension field. Second, it is not obvious how one can evaluate elements of $B_{p,\infty}$ as endomorphisms. The first issue is solved by cutting up the ideal into smaller chunks, such that for every chunk the kernel is defined over a small extension field.

The second issue is more complicated, furthermore, there is a difference between the original SQISign construction [14] and the improved version [15]. First we recall the original construction. Let $\alpha \in \mathcal{O}$ and we would like to evaluate it on certain points. Now α corresponds to endomorphism of E which (by abuse of notation) we will also denote by α. The order $\mathcal{O}_0$ has the special property that every element of $\mathcal{O}_0$ can be evaluated in a natural fashion (as i corresponds to the automorphism $(x, y) \mapsto (-x, iy)$ and j corresponds to the Frobenius endomorphism). If α is in the Eichler order $\mathcal{O}_0 \cap \mathcal{O}$, then we can evaluate it in $\mathcal{O}_0$. In order to evaluate it on $\mathcal{O}$ we need to use KLPT to translate it to E.

In the improved version [14] they do not need to evaluate all endomorphisms of the Eichler order $\mathcal{O}_0 \cap \mathcal{O}$, just a well-chosen one. This is accomplished with the SpecialEichlerNorm algorithm which provides a $\beta \in \mathcal{O}$ with a specific norm and some extra property. Now these special elements are used throughout the ideal to isogeny translation algorithm and KLPT is no longer needed. This results in a considerable speed-up, the main reason being that that a much better prime p can be chosen this way.

We have omitted many details from [14] and [15] but we will recall three algorithms in its entirety here as they will be relevant to our attack: RepresentInteger, StrongApproximation, and SpecialEichlerNorm.

Algorithm 1 $\mathsf{RepresentInteger}_{\mathcal{O}_0}(\mathsf{M})$ [15]

Require: $M \in \mathbb{Z}$ such that $M > p$.
Ensure: $\gamma = x + yi + zj + tik$ with $n(\gamma) = M$.
1: Set $m = \left\lfloor \sqrt{\frac{M}{p(1+q)}} \right\rfloor$ and sample random integers $z, t \in [-m, m]$.
2: Set $M' = M - pf(z, t)$.
3: **if** Cornacchia$(M') = \perp$ **then**
4: Go back to Step 1.
5: **else**
6: $x, y \leftarrow$ Cornacchia(M').
7: **end if**
8: $\gamma = (x + iy + j(z + it)$.
9: **return** γ.

Algorithm 2 StrongApproximation [14]

Require: Prime N, l a non quadratic residue modulo N, and $C, D \in \mathbb{Z}$.
Ensure: $\mu = \lambda \cdot \mu_0 + N \cdot \mu_1$, with $\mu_0 = j \cdot (C + \omega D)$, $\mu_1 \in \mathcal{O}_0$ such that $n(\mu) = l^{e_1}$ for some $e_1 \in \mathbb{N}$.
1: Select $e_1 \leq p \cdot N^4$ and adjust the parity s.t $\frac{l^e}{p \cdot (C^2 + qD^2)}$ is a quadratic residue mod N. We denote its square root as λ.
2: Select z, t such that $l^e - p \cdot f(\lambda C + Nz, \lambda D + Nt) = 0 \mod N^2$.
3: Set $M = \frac{l^e - p \cdot f(\lambda C + Nz, \lambda D + Nt)}{N^2}$.
4: **if** Cornacchia$(M) = \perp$ **then**
5: Go back to Step 2.
6: **else**
7: $x, y \leftarrow$ Cornacchia(M).
8: **end if**
9: **return** $\mu = \lambda j(C + D\omega) + N(x + \omega y + j(z + \omega t))$.

Algorithm 3 SpecialEichlerNorm [15]

Require: $\mathcal{O}$ a maximal order and K a left $\mathcal{O}$-ideal of norm l.
Ensure: $\beta \in \mathcal{O} \backslash (\mathbb{Z} + K)$ of norm dividing T^2.
1: Compute $I = I(\mathcal{O}_0, \mathcal{O})$.
2: Set $L =$ RandomEquivalentPrimeIdeal(I), $N = n(L)$ and compute α such that $L = I\alpha$.
3: Compute $K' = \alpha^{-1} \cdot K \cdot \alpha$.
4: Compute $(C : D) =$ EichlerModConstraint$(L, 1)$.
5: Enumerate all possible solutions of $\mu =$ FullStrongApproximation(N, C, D) until $\mu \notin \mathbb{Z} + K'$. If it fails, go back to Step 2.
6: **return** $\beta = \alpha\mu\alpha^{-1}$.

2.4 Side-channel attacks

Even though the mathematical 'hard' problem at the core of a cryptographic primitive ensures theoretical security, its practical implementation often leaks potential intermediate information. An adversary can observe and exploit these side-channel leaks. There are a few different classes of such side-channel attacks, some target the physical characteristics of a system while others measure changes in the timing or power consumption associated with the operations being executed. The most vulnerable are those algorithms with conditional branches dependent on secret inputs.

Attacks on elliptic curve-based cryptosystems usually target the point-doubling operation or general scalar multiplications. While SQISign also has elliptic curve computations and so could have such vulnerabilities, what makes it interesting is that there are also other possibilities of attack scenarios due to additional quaternion arithmetic. We explore one such scenario: a non-constant time Euclid's GCD algorithm. Many of its versions have been found to be 'leaky' in

side-channel literature, [1,4,3,2]. This algorithm can also be terminated 'half-way', leading to a sub-quadratic algorithm known as 'half-GCD'. SQISign uses the half-GCD algorithm inside one of its subroutines, the Cornacchia's algorithm to express an integer as a sum of two squares. In this paper, we show how an adversary can guess the signing key of SQISign if they are able to exploit the side-channel vulnerabilities of the Euclid's algorithm. As timing-resistant replacements, we also provide two different countermeasures.

2.5 Algorithms: Cornacchia and Euclid

Cornacchia's algorithm states that, for two co-prime integers m and d, the solution of $x^2 + dy^2 = m$, in coprime integers x and y (if any) is given by the Euclid's algorithm applied to the pair (x_0, m) where x_0 is any root of $x^2 \equiv -d \pmod{m}$. The algorithm stops when in the successive sequence (r_n) of remainders, we find an r_k satisfying the relation, $r_k^2 < m \leq r_{k-1}^2$. In practical implementations, this translates to employing the 'half-GCD' algorithm. The solution is then, $\left(x = r_k, y = \sqrt{\frac{m-r_k^2}{d}}\right)$ if y^2 is a square integer, otherwise the process is repeated with another modular square root x_0' until all such roots get exhausted. SQISign implements a version of the Cornacchia's algorithm for $d = 1$. We mention this algorithm in alg. 4.

Algorithm 4 Cornacchia(m) [14]

Require: $m \in \mathbb{Z}$
Ensure: $x, y \in \mathbb{Z}$ such that $x^2 + d \cdot y^2 = m$. ▷ $d = 1$ in SQISign
1: Compute $u = \sqrt{-d} \pmod{m}$.
2: Compute $r_k = \mathsf{halfgcd}(u, m)$.
3: Check, $\frac{\sqrt{m-r_k^2}}{d} \in \mathbb{Z}$.
4: **if** False **then**
5: go to step 1.
6: **else**
7: $x = r_k, y = \frac{\sqrt{m-r_k^2}}{d}$.
8: **end if**
9: **return** x, y

The sub-routine halfgcd in alg. 4 actually uses the Euclid's algorithm but terminates it 'half-way' when the internal condition of $r_k < \sqrt{m}$ is met, as given in alg. 5.

The Euclid's algorithm is a sequence of recursive steps used to compute the greatest common divisor (GCD) of two integers. For two integers $a, b \in \mathbb{Z}$ and assuming that $a > b$, it can be visualised as a sequence of linear combinations of successive quotients q_i and remainders r_i such that, $r_{i+1} = r_{i-1} - q_{i+1} r_i$. If after certain N steps of the algorithm, the remainder $r_N = 0$ then the last non-zero remainder r_{N-1} is the GCD of a and b, that is, $r_{N-1} = gcd(a, b)$.

Algorithm 5 $\mathsf{halfgcd}(m, u)$ [14]

Require: $m, u \in \mathbb{Z}^2$.
Ensure: $a \in \mathbb{Z}$.
1: Set $l = \lfloor \sqrt{m} \rfloor$.
2: Set $a, b = m, u$
3: **while** $a > l$ **do**
4: Compute $r = a \pmod b$.
5: Set $a, b = b, r$.
6: **end while**
7: **return** a

2.6 Lattices and Lagrange reduction

A lattice L of dimension r is a maximal discrete subgroup of $\mathbb{R}^n$. In other words, $L = \{\sum_{i=1}^{r} c_i \mathbf{v_i} : c_i \in \mathbb{Z},\ i \in \{1, 2, \cdots, r\}\}$. Thus a lattice consists of all $\mathbb{Z}$-linear combinations of linearly independent vectors $\mathbf{v_i} \in \mathbb{R}^n$. The vectors $\mathbf{v_i}$ comprise a 'basis', β of the lattice. A basis β can also be represented by a matrix B of row vectors such as,

$$B = \begin{bmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_r} \end{bmatrix} = \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ v_{21} & \cdots & v_{2n} \\ \vdots & \ddots & \vdots \\ v_{r1} & \cdots & v_{rn} \end{bmatrix}$$

A lattice L usually has an infinite number of bases. If $\{\mathbf{v_1}, \cdots, \mathbf{v_r}\}$ is a basis then another basis would be $\{\mathbf{v_1}, \cdots, \mathbf{v_{i-1}}, \mathbf{v_i} + k\mathbf{v_j}, \mathbf{v_{i+1}}, \cdots, \mathbf{v_r}\}$, where $i \neq j$ and $k \in \mathbb{Z}$. The volume of the lattice L is given by $\mathrm{V}(L) = \sqrt{\det(G)}$, where $G = B \cdot B^t \in \mathbb{R}^{r \times r}$ is the associated 'Gram matrix'. Transitioning between basis matrices, say B and B' is usually equivalent to multiplying one of the basis matrices with an unimodular matrix M over $\mathbb{Z}$, i.e., $B' = M \cdot B$. The volume of the lattice however, remains invariant, $\sqrt{\det(G)} = \sqrt{\det(G')}$. A lattice basis transition is necessary, for example, when one wants to find a "nice" basis for the lattice. A measure for how "nice" the basis is could be found in the 'orthogonality defect' of a basis. The more orthogonal (lesser defect) the vectors are, the "nicer" is the basis. The orthogonality defect is defined as,

$$\mathrm{def}(\mathbf{v_1}, \cdots, \mathbf{v_r}) = \frac{\|\mathbf{v_1}\| \cdots \|\mathbf{v_r}\|}{\mathrm{Vol}(L)} \geq 1.$$

Lattice basis reduction refers to techniques that are used to transform a given lattice basis into a "nice" lattice basis consisting of vectors that are short and close to orthogonal. This means that algorithms for lattice reduction aim to reduce the orthogonality defect of the starting basis as much as possible. Reduction of two dimensional lattice bases in $\mathbb{R}^2$ was given by Lagrange and Gauss. The LLL algorithm [24] is used for higher dimension lattices. It generalises the Lagrange-Gauss algorithm and uses the Gram-Schmidt orthogonalisation process. Here, we make use of the algorithm by Lagrange-Gauss 6 which we

will call 'Lagrange Reduction' from hereafter. An ordered basis $\mathbf{v_1}, \mathbf{v_2} \in \mathbb{R}^2$ is called Lagrange-reduced if $\|\mathbf{v_1}\| \leq \|\mathbf{v_2}\| \leq \|\mathbf{v_2} + k\mathbf{v_1}\|$, $\forall k \in \mathbb{Z}$. This algorithm is closely related to Euclid's algorithm and we give a brief of the discussion by [19]. For $x, y \in \mathbb{Z}$ the Euclid's algorithm produces a sequence of integers r_i, s_i, t_i such that $xs_i + yt_i = r_i$ such that $|r_i t_i| < |x|$ and $|r_i s_i| < |y|$. The initial values are: $r_{-1} = x, r_0 = y, s_{-1} = 1, s_0 = 0, t_{-1} = 0, t_0 = 1$. Then the lattice L generated by the basis matrix,

$$B = \begin{bmatrix} 0 & y \\ 1 & x \end{bmatrix} = \begin{bmatrix} s_0 & r_0 \\ s_{-1} & r_{-1} \end{bmatrix}$$

contains vectors of the form $(s_i, r_i) = (t_i, r_i) \cdot B$. These vectors are shorter with smaller orthogonality defect than the starting basis vectors of the lattice.

Algorithm 6 LagrangeReduction(u, v)

Require: $u, v \in \beta(L) \subseteq \mathbb{Z}^2$.
Ensure: u, v two 2-D vectors such that
1: $B1 = \|u\|^2$.
2: Compute $\mu = \lfloor \frac{(u|v)}{B1} \rfloor$.
3: Compute $v = v - \mu \cdot u$.
4: $B2 = \|v\|^2$.
5: **while** $B2 < B1$ **do**
6: Swap u and v.
7: $B1 = B2$
8: Compute $\mu = \lfloor \frac{(u|v)}{B1} \rfloor$.
9: Compute $v = v - \mu \cdot u$.
10: $B2 = \|v\|^2$.
11: **end while**
12: **return** u, v

3 Vulnerabilities of Cornacchia and retrieval of the signing key

In this section we will first give an outline of the side-channel vulnerabilities of Cornacchia, and more specifically the non-constant time Euclid's algorithm. We will then describe a method to retrieve the endomorphism ring of E (the secret key in SQISign) if one has a way of obtaining the inputs and hence the outputs of Cornacchia.

In the context of SQISign the relevant Cornacchia variant is when one wants to represent an integer m as $x^2 + y^2$. In actual implementation, the authors use an 'extended' version of the algorithm which we mention in alg 7. It ensures that m is almost always a prime and definitely odd.

Algorithm 7 ExtendedCornacchia(M) [14]

Require: $M \in \mathbb{Z}$.
Ensure: $x, y \in \mathbb{Z}$ such that $x^2 + y^2 = m$.
1: $h_1 = 2^{\nu_2} \cdot 5^{\nu_5} \cdot 13^{\nu_{13}} \cdot \cdots \cdot 101^{\nu_{101}}$ ▷ v_p on an integer is it's p-valuation
2: $h_3 = 3^{\nu_3} \cdot 7^{\nu_7} \cdot \cdots \cdot 83^{\nu_{83}}$ ▷ all 3 (mod 4) primes until 101
3: **if** $\gcd(M, h_3) == 1$ **then**
4: Compute $m = \frac{M}{h_1}$.
5: Compute $g = \gcd(m, h_1)$.
6: **while** $g \neq 1$ **do**
7: $m = m/g$
8: $g = \gcd(g, (m \pmod g))$
9: **end while**
10: Compute $x, y =$ Cornacchia(m).
11: **return** x, y
12: **end if**

Let us assume that an attacker is able to observe and then analyse the variations in the number of times the division step runs within the halfgcd sub-routine in alg 5, which reveals m. From experimental observations it seems that in large number of instances of the signing algorithm, when M is more than 64-bit long, either $M = m$ or, $M = 2^{\nu_2} m$. Next, we show how one can retrieve the output of StrongApproximation. Fig. 1 gives an outline of the key recovery procedure in correspondence with the algorithm hierarchy within SQISign's signing routine. For easy visualisation, we denote the secret endomorphism ring $\mathcal{O}$ by 's'.

Theorem 1. *Suppose we have a way of retrieving outputs and inputs of* Cornacchia*. Then one can obtain the output of* StrongApproximation *in KLPT if the output size l^e is known.*

Proof. Since we have access to Cornacchia inputs and outputs, we know M and x, y such that $M = x^2 + y^2$. Then one has the equality

$$MN^2 = l^e - p((\lambda C + NZ)^2 + (\lambda D + Nt)^2) \tag{1}$$

and hence, $MN^2 \equiv l^e \pmod p$. Since M and l^e are known, we have two choices for N modulo p. In KLPT, $N \approx \sqrt{p}$ which gives us the exact value of N. Indeed, every residue has 2 square roots modulo p, namely some a and the other $p - a$. This implies that the other square root of N^2 will be bigger than $p/2$. Equation 1 implies that we know $(\lambda C + NZ)^2 + (\lambda D + Nt)^2$. From this one can solve the equation $(\lambda C + NZ)^2 + (\lambda D + Nt)^2 = x_0^2 + y_0^2$ and compute all solutions x_0, y_0. Usually there should only be a few solutions here, so by testing we may assume that we have found $\lambda C + NZ$ and $\lambda D + Nt$. The output of StrongApproximation is $\mu = Nx + Nyi + (\lambda C + NZ)j - (\lambda D + Nt)ij$, and thus, we have found μ as we now know every coordinate of μ.

Proposition 1. *Suppose we have a way of retrieving outputs and inputs of* Cornacchia*. Then we can obtain a small set of valid outputs (amongst which*

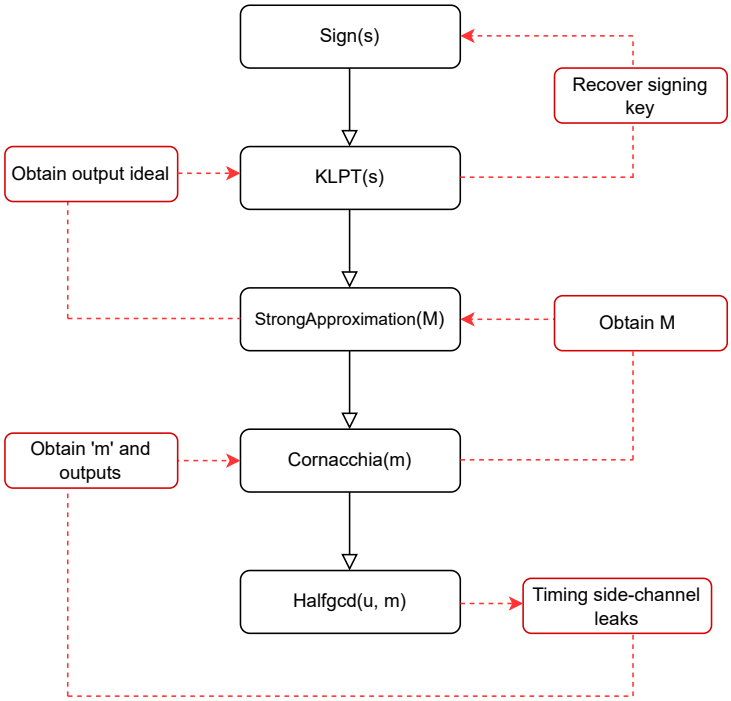

Fig. 1: Signing key recovery

is the actual output) γ *of* $\mathsf{RepresentInteger}_{\mathcal{O}_0}$ *in polynomial time if either of the following conditions is satisfied:*

- $n(\gamma)$ *is of the form* Nl^e *where* l, e *are known and* $N < p$.
- $n(\gamma) = M < \log(p)^c$ *for some constant* c

Proof. First one can obtain m or M as the input of Cornacchia. From this we know x, y along with M such that $x^2 + y^2 = M$. Now we split our algorithm into two cases depending on which condition is satisfied.

If the first condition is satisfied, then we know that $n(\gamma) = M = Nl^e$ where l^e is known and $N < p$. We also know that $M = M' - p(z^2 + t^2)$ where we know p but don't know z and t. This implies that we know M modulo p. Now as $M = Nl^e$ and l^e is known and coprime to p, we can obtain N modulo p by modular inversion. Since $N < p$, we have obtained N exactly which implies that we have obtained M as well. This in turn implies that $z^2 + t^2$ is known and we can compute z, t with Cornacchia. There are usually not too many solutions to choose from and all the possible solutions can be computed from factoring $z^2 + t^2$ (in practice this is a small number).

If the second condition is satisfied, then we can proceed in a similar fashion. Here we just need to guess M. Since $M < \log(p)^c$, one would get only $\log(p)^c$ possible outputs. In this setting $z^2 + t^2$ is small by design.

Theorem 1 and Proposition 1 are the main building blocks of our attacks. We will provide two key recovery methods, one on the original SQISign construction [14] and one on the newer version [15]. First we focus on the original construction.

Theorem 2. *Suppose we have a way of retrieving outputs and inputs of* Cornacchia*. Then one can retrieve the output ideal of the KLPT algorithm.*

Proof. The KLPT algorithm, as mentioned before (and described in [22]) has three main steps. First computing γ which is the output of the $\mathsf{RepresentInteger}_{\mathcal{O}_0}$ algorithm. Then computing a μ_0 such that $(\mathcal{O}_0\gamma)\mu \equiv I \pmod{N\mathcal{O}_0}$ and then lifting μ_0 to μ whose norm is l^e. Then $\gamma\mu$ will be an element of I of norm Nl^e which provides an equivalent ideal I J of norm l^e. Theorem 1 implies that we can obtain μ and Proposition 1 implies that we can obtain γ. Putting these together we obtain $\gamma\mu$ which is exactly needed to get the output of KLPT.

Remark 1. Proposition 1 might not be strictly necessary in certain contexts if the algorithm $\mathsf{RepresentInteger}_{\mathcal{O}_0}$ is performed deterministically as one can just recomputes the algorithm offline. The reason is that 1 also provides N (not just μ), hence if l^e is chosen deterministically one does not need to use the Cornacchia oracle again. Actually, if N is a prime congruent to 1 modulo 4, often one just chooses $e_0 = 0$.

Corollary 1. *Suppose we have a way of retrieving outputs and inputs of* Cornacchia*. Then we can get the signing key in SQISign [14] in polynomial time.*

Proof. We focus on [14, Algorithm 9] which provides the ideal-to-isogeny translation. In Step 2, KLPT is used to obtain an equivalent ideal J which is a connecting ideal between $\mathcal{O}_0$ and $\mathcal{O}$ where $\mathcal{O}$ is the endomorphism ring of the public curve. Theorem 2 implies that we can get J which immediately reveals $\mathcal{O}$ as the right order of J is isomorphic to $\mathcal{O}$.

Theorem 2 is not directly applicable to the new SQISign version [15] as in the ideal-to-isogeny translation algorithm KLPT is no longer used. Instead of computing with entire endomorphism ring, the algorithm SpecialEichlerNorm is called which computes a well chosen endomorphism of the appropriate maximal order. In the next theorem we show how to apply 1 to obtain the signing key.

Theorem 3. *Suppose we have a way of retrieving outputs and inputs of* Cornacchia*. Then we can retrieve the signing key in the improved version of SQISign [15] in polynomial time.*

Proof. We target Step 5 of SpecialEichlerNorm which is a call to StrongApproximation algorithm. Theorem 1 implies that we can obtain μ using our Cornacchia approach. Furthermore, the proof of Theorem 1 implies that we also retrieve $n(L) = N$ as $N < p$. Our goal is to compute generators for the ideal L as then the right order of L will be $\mathcal{O}$ that reveals the endomorphism ring of the public curve (as the first input of SpecialEichlerNorm is the order $\mathcal{O}$). Now $\mu \in \mathbb{Z} + L$, so μ alone is not enough to retrieve L. Since $\mu \in \mathcal{O}_0$ one can write it as a 2×2 matrix

with entries from $\mathbb{Z}/N\mathbb{Z}$ via the explicit isomorphism $\mathcal{O}_0/N\mathcal{O}_0 \cong M_2(\mathbb{Z}/N\mathbb{Z})$. Elements in L have the property that when they are written as 2×2 matrices, they are not invertible as their norms are divisible by N. Now $\mu = \lambda + \sigma$ where $\lambda \in \mathbb{Z}$ and $\sigma \in L$ hence one needs to figure out the value of λ. Since $N\mathcal{O}_0$ is contained in L, actually, λ is only determined modulo N. Also, since elements in L correspond to matrices that are not invertible, $\mu - \lambda$ is not an invertible matrix which is equivalent to λ being an eigenvalue of μ. Eigenvalues can be computed efficiently as N is a prime number. In general μ ha two eigenvalues which provides two possible choices λ_1 and λ_2 for λ. Elements in L correspond to a set of matrices in $M_2(\mathbb{Z}/N\mathbb{Z})$ whose kernel contains a particular cyclic subgroup of $(\mathbb{Z}/N\mathbb{Z})^2$ (or equivalently a point in $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$). Hence for both λ_i's we compute the kernel of $\mu - \lambda_i$ and compute the corresponding ideal L_i. By computing right orders we get two candidates $\mathcal{O}_1$ and $\mathcal{O}_2$ for $\mathcal{O}$. Then we compute the supersingular elliptic curves corresponding to these orders and choose the one whose j-invariant matches that of the public curve. The last step can be accomplished in polynomial time and is quite practical using the methods of SQISign or that of [17].

We note here that the aforementioned theoretical key recovery procedure would work only if a timing-based side-channel attack gives full information about the inputs and outputs of Cornacchia. In practice, a timing-attack alone might not be enough, it may need to be merged with other forms of side-channel attacks such as a power analysis. We do not claim to have the whole picture of such a feasible attack. However, in the security community we do not wait for the first attack to happen. We proactively develop secure algorithms that can already be made resilient to all forseeable future attacks as well. It thus remains an open problem to device a full-scale practical attack against the non-constant time elements of the signing algorithm.

4 Timing attack-resistant alternatives

It is clear from our discussion in the previous section that one needs to replace Cornacchia or look for constant-time algorithms to achieve a secure implementation of SQISign.

4.1 Lattice reduction

In SQISign, Cornacchia is used to write a number as a sum of two squares. In general this can be a hard algorithmic problem as it is essentially equivalent to factoring the number. In SQISign this is circumvented by iterating until the number to be expressed as a sum of two squares is prime. Here we describe an alternative way to write a prime number M congruent to 1 modulo 4 as a sum of two squares. First one computes an integer u such that $u^2 \equiv -1 \pmod{M}$. This can be done efficiently (e.g., choosing a random element and raising it to the $(M-1)/4$-th power). Then one can look at the lattice L generated by $(1, u)$ and $(0, M)$. This lattice has two properties:

– Every vector (a, b) in this lattice has the property that $a^2+b^2 \equiv 0 \pmod{M}$.
– The determinant of the lattice is M.

Now, Minkowski's theorem implies that the shortest non-zero vector in this lattice is shorter than $2M$, hence its length is exactly M. Thus solving $x^2+y^2 = M$ amounts to finding the shortest vector in this lattice which can be accomplished efficiently with Lagrange reduction (alg 6). In order to have a timing attack resistant algorithm for Cornacchia, we re-randomize the starting basis of L by multiplying the matrix of L by a random matrix of determinant 1. Since the matrix has determinant 1, it has no affect on the solution space. With that, the number of executions of the steps from 5-11 in alg. 6 no longer depend directly on the input basis elements but their random scalar combinations. The worst-case complexity of Lagrange reduction has been studied extensively by [23] and [34] which also discusses the worst-case input types.

Algorithm 8 (Randomised)LatticeReduction(M, d)

Require: M, d two integers. ▷ $d = 1$ in SQISign.
Ensure: x, y such that $x^2 + d \cdot y^2 = 0 \mod M$
1: Set $u^2 = -d \mod M$.
2: Generate r_0, r_1 two randoms integers.
3: Generate the lattice $L = \begin{bmatrix} 1 & 0 \\ u & M \end{bmatrix}$
4: Generate $S = \begin{bmatrix} 1 & r_0 \\ r_1 & r_0 r_1 + 1 \end{bmatrix}$ ▷ Or any random matrix of determinant 1.
5: Compute $L_0 = S \cdot \begin{bmatrix} 1 \\ u \end{bmatrix}$ and $L_1 = S \cdot \begin{bmatrix} 0 \\ M \end{bmatrix}$
6: Compute $l_0, l_1 =$ LagrangeReduction(L_0, L_1).
7: **return** $x = l_0[0]$, $y = l_0[1]$

4.2 Constant-time half-GCD

Another way of mitigating the attack lies in the use of a constant-time implementation of the halfgcd algorithm. A work by [6] proposes certain tweaks such that the iterations within the GCD algorithm no longer depend on the inputs but on auxiliary parameters. It follows an algorithmic flow similar to the Stein's [33] binary-GCD algorithm. [6] introduces a function called the 'divstep', which is the actual constant-time element in the algorithm. The function, $divstep : \mathbb{Z} \times \mathbb{Z}_2^* \times \mathbb{Z}_2 \to \mathbb{Z} \times \mathbb{Z}_2^* \times \mathbb{Z}_2$ is defined as,

$$divstep(\delta, a, b) = \begin{cases} (1+\delta, b, (b-a)/2), & \text{if } \delta > 0 \text{ and } b \text{ is odd} \\ (1-\delta, b, (b+(b \pmod 2))a)/2), & \text{otherwise} \end{cases}$$

In this GCD implementation, the auxiliary parameter δ depends on the maximum of the two input sizes, k, i.e., for two inputs a and b, $|a| < 2^k, |b| < 2^k$. The algorithm is constant-time if k is constant. It then calculates an integer n under

a special function of k, $\mathsf{iterations}(k)$. This integer n serves as an upper bound of the number of iterations, [6, Theorem 11.2]. It ensures that whenever *divstep* is executed n times with initial inputs (a, b), the sequence of intermediate values a_i and b_i are updated in a way such that, $a_n \to \pm gcd(a, b)$ and, $b_n \to 0$.

However this algorithm cannot be trivially ported into a half-GCD algorithm as we demonstrate with the following two examples:

– Let us assume that we need to apply Cornacchia's algorithm to two integers, $M = 7349, u = 2061$. Note that $\lfloor\sqrt{M}\rfloor = 85$. Starting with $a_n = M, b_n = u$, a half-GCD algorithm would terminate as soon as it encounters an $a_i < 85$. In such a scenario, the Euclid's algorithm stops at the first such a_i, namely, $a_5 = 82$ such that the square root of $M - a_i$ is also an integer. Hence, $(82, 25)$ are valid solutions of Cornacchia. However, the constant GCD algorithm oscillates between positive and negative values. Although it comes across $b_{24} = 25$, the values of the preceding steps are already $b_{25} = 50$ or $|b_{31}| = 48$. If we were to use Cornacchia's bound-check naively, this algorithm would wrongly terminate at $|b_{31}|$ or b_{25} and Cornacchia would then begin a second run with a new modular square root.
The constant-time GCD algorithm [6] produces the two sequences, $a_{i+1}, b_{i+1} = \text{divstep}(a_i, b_i)$ for $i \in [0, n]$ as shown in fig 2. The steps that the usual Euclidean algorithm follows is given in fig 3.

a	7349	2061	2061	2061	−661	−661	−661	−661	−661	−661	−209	−209	−209	−209	−209	−209
b	2061	−2644	−1322	−661	−1361	−1011	−836	−418	−209	226	113	−48	−24	−12	−6	−3
n	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27

contd..

a	−3	−3	−3	−3	−3	−3	−3	1	1	1	1	1	···	1
b	103	50	25	11	4	2	1	2	1	1	1	0	···	0
n	26	25	24	23	22	21	20	19	18	17	16	15	···	1

Fig. 2: Constant-time gcd(7349, 2061)

a	7349	2061	1166	895	271	82	25	7	4	3
b	2061	1166	895	271	82	25	7	4	3	1
n	10	9	8	7	6	5	4	3	2	1

Fig. 3: Euclid's gcd(7349, 2061)

– Next, we apply both GCD algorithms to another pair of integers, $(61, 11)$, where 11 is a modular square root of 61. The Euclid's algorithm would terminate at the very first iteration, $61 = 11 \cdot 5 + 6$, because $6^2 < 61$. It is easy to see that $(6, 5)$ is also a valid solution of the Cornacchia's equation, $x^2 + y^2 = 61$. In contrast however, the constant-time GCD algorithm does not come across either of the values, 6 or 5, as shown in fig 4. In this case again, it would have to re-run the algorithm with another possible modular square root of 61.

a	61	11	11	−7	−7	−7	−7	−7	−1	−1	−1	···	−1
b	11	−25	−7	−9	−8	−4	−2	−1	3	1	0	···	0
n	22	21	20	19	18	17	16	15	14	13	12	···	1

Fig. 4: Constant-time gcd(61, 11)

Therefore, taking inspiration from [6] and applying the iteration bound of n, we propose a constant-time half-GCD algorithm in 9. We take into account two sequences and make them converge to specific values. Our sequences of intermediate values are n-terms long, with n being calculated based on the size of the input as in [6]. We compute the same intermediate values as the usual Euclidean algorithm. In fact, we can make use of Cornacchia's bound check, l (step 1, alg. 9) to control the first sequence $(a_i)_{i\in\{0,n\}}$ in our algorithm in such a way that it converges to the half-GCD by successive subtraction of the larger integer by the smaller one. The value of the variable 'sign' (step 8, alg. 9) becomes 0 as soon as the algorithm encounters a value less than l. The second sequence $(b_i)_{i\in\{0,n\}}$ then converges towards 0, it reaches 0 when the value of a_i at a certain step i is approximately less than half of the initial value, thus freezing the value of the sequence $(a_i)_{i\in\{0,n\}}$ at the half-GCD. It is this sequence that helps us achieve constant time, as past a certain point i in the iterations, the intermediate values no longer change till the end of n iterations.

The rationale behind introducing the variable ν in Alg. 9, step 10, can be argued as follows: without ν, step 10 would include many subtractions which could result in very large worse-case bounds for the algorithm. As a way of optimizing this step, we can merge multiple subtractions by $\min(a, b, c)$ into one subtraction by $2^{\nu} \cdot \min(a, b, c)$. If the minimum and the maximum value have the same number of bits then we set $\nu = 0$, else, ν is set to one less than the bit difference of the maximum and the minimum values.

5 Performance results

We compiled our implementation of C-halfgcd in C programming language using gcc-11.3 with optimization flags -O3 and measured computation time using a

Algorithm 9 C-halfgcd(M, u)

Require: M, u, d such that $u^2 = -d \mod M$
Ensure: x, y such that $x^2 + d \cdot y^2 = M$
1: $l = \lfloor \sqrt{M} \rfloor$
2: $k = \max(nbit(M), nbit(u))$
3: $n = \mathsf{iterations}(k)$
4: $a, b = M, u$
5: **while** $n > 0$ **do**
6: $c = \max(a, b)$
7: $k_c = \mathrm{nbit}(c)$
8: $\mathrm{sign} = ((l - c)\&(1 \ll k_c)) \gg k_c$
9: $b = \min(a, b, c) \cdot \mathrm{sign}$
10: $a = \max(a, b, c) - \min(a, b, c) \ll \nu$ ▷ $\nu = 0$, or, $nbit(\max) - nbit(\min) - 1$.
11: $n = n - 1$
12: **end while**
13: $x = a,\ y = \sqrt{\frac{M - x^2}{d}}$
14: **return** x, y

single core of an Intel(R) Core(TM) i7-1260P processor running at 4.70GHz on an Lenovo ThinkPad T14s laptop with Ubuntu 22.04 operating system. We remark that a fair comparison of the (randomised) lattice reduction method, the constant half-GCD and the GP-PARI-based implementation of the half-GCD algorithm in SQISign would not be precise. First, the lattice reduction algorithm is not constant-time: it simply removes input-dependency and also needs to generate two random integers. Moreover, we also cannot compare our techniques with the current SQISign implementation as it uses the GP-PARI library extensively while ours don't.

We implement C-halfgcd for different size of primes ranging from 240-bit to 400-bit (which are actual SQISign parameters for NIST security level-I). To verify that our implementation is indeed constant-time, we make use of the 'ctgrind' library and the 'Valgrind' analysis tool. In tab. 1, we present the timing results with respect to the number of cycle counts (cc) with their mean and standard deviation for C-halfgcd with respect to different prime sizes.

prime size range (bits)	**mean (cc)**	**sd (cc)**
$240 - 250$	292250	24894
$250 - 260$	345370	64602
$375 - 395$	473959	69324

Table 1: Cycle count for C-halfgcd

6 Conclusion

In this paper, we showed how the current implementation of SQISign could aid an adversary to recover the secret signing key if it is subjected to side-channel attacks. We presented a complete polynomial-time key recovery method that exploits the vulnerability of the very simple, yet widely-used Euclid's algorithm. The non-constant time branches within the algorithm could give away important information about its inputs to a malicious observer. Then, only a few algebraic manipulations would suffice to reveal the secret key. The complex structure of the scheme necessitates many algorithmic as well as implementational optimisations. However, it is interesting, for example, to investigate if and how these optimisation tricks could induce side-channel vulnerabilities.

Additionally, we also proposed two timing-attack resilient alternatives that could replace the non-constant algorithms. The first one is based on lattice reduction techniques in two dimensions. We suggested randomisation of the starting lattice basis to eliminate input-dependency. The second one is a constant-time half-GCD algorithm. The constant-time feature of the algorithm comes from setting an upper bound on the iterations based only on the maximum of the input sizes. We also analysed the performance of our algorithm with a C implementation and verified that it is constant-time using standard tools. A future direction would to derive better, more concrete bounds that would still ensure correct termination. In fact, coming up with a better bound could significantly improve the complexity of the algorithm.

With this work, we aim to motivate further cryptanalysis of SQISign so that, endowed with an attack-resistant implementation, the scheme can realise its full potential in the post-quantum world.

7 Acknowledgement

This work was supported in part by the State Government of Styria, Austria – Department Zukunftsfonds Steiermark.

References

1. Aciiçmez, O., Gueron, S., Seifert, J.: New branch prediction vulnerabilities in openssl and necessary software countermeasures. In: Galbraith, S.D. (ed.) Cryptography and Coding, 11th IMA International Conference, Cirencester, UK, December 18-20, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4887, pp. 185–203. Springer (2007). https://doi.org/10.1007/978-3-540-77272-9_12, https://doi.org/10.1007/978-3-540-77272-9_12
2. Aldaya, A.C., Brumley, B.B.: When one vulnerable primitive turns viral: Novel single-trace attacks on ECDSA and RSA. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(2), 196–221 (2020). https://doi.org/10.13154/tches.v2020.i2.196-221, https://doi.org/10.13154/tches.v2020.i2.196-221

3. Aldaya, A.C., Sarmiento, A.C., Sánchez-Solano, S.: SPA vulnerabilities of the binary extended euclidean algorithm. J. Cryptogr. Eng. **7**(4), 273–285 (2017). https://doi.org/10.1007/s13389-016-0135-4, https://doi.org/10.1007/s13389-016-0135-4
4. Aravamuthan, S., Thumparthy, V.R.: A parallelization of ECDSA resistant to simple power analysis attacks. In: Paul, S., Schulzrinne, H., Venkatesh, G. (eds.) Proceedings of the Second International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE 2007), January 7-12, 2007, Bangalore, India. IEEE (2007). https://doi.org/10.1109/COMSWA.2007.382592, https://doi.org/10.1109/COMSWA.2007.382592
5. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The sphincs+ signature framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2129–2146. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363229, https://doi.org/10.1145/3319535.3363229
6. Bernstein, D.J., Yang, B.: Fast constant-time gcd computation and modular inversion. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(3), 340–398 (2019). https://doi.org/10.13154/tches.v2019.i3.340-398, https://doi.org/10.13154/tches.v2019.i3.340-398
7. Beullens, W., Kleinjung, T., Vercauteren, F.: Csi-fish: efficient isogeny based signatures through class group computations. In: Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I. pp. 227–247. Springer (2019)
8. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of csidh. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30. pp. 493–522. Springer (2020)
9. Castryck, W., Decru, T.: An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive pp. Paper–2022 (2022)
10. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: Csidh: an efficient post-quantum commutative group action. 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 24. pp. 395–427. Springer (2018)
11. Couveignes, J.M.: Hard homogeneous spaces. Preprint at https://eprint. iacr. org/2006/291 (1999)
12. De Feo, L., Dobson, S., Galbraith, S.D., Zobernig, L.: Sidh proof of knowledge. In: Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part II. pp. 310–339. Springer (2023)
13. De Feo, L., Galbraith, S.D.: Seasign: compact isogeny signatures from class group actions. In: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38. pp. 759–789. Springer (2019)
14. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: 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 26. pp. 64–93. Springer (2020)

15. De Feo, L., Leroux, A., Wesolowski, B.: New algorithms for the deuring correspondence: Sqisign twice as fast. Cryptology ePrint Archive (2022)
16. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(1), 238–268 (2018). https://doi.org/10.13154/tches.v2018.i1.238-268, https://doi.org/10.13154/tches.v2018.i1.238-268
17. Eriksen, J.K., Panny, L., Sotáková, J., Veroni, M.: Deuring for the people: Supersingular elliptic curves with prescribed endomorphism ring in general characteristic. Cryptology ePrint Archive (2023)
18. Feo, L.D., Fouotsa, T.B., Kutas, P., Leroux, A., Merz, S.P., Panny, L., Wesolowski, B.: Scallop: Scaling the csi-fish. In: Public-Key Cryptography–PKC 2023: 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7–10, 2023, Proceedings, Part I. pp. 345–375. Springer (2023)
19. Galbraith, S.D.: Lattice basis reduction, p. 347–365. Cambridge University Press (2012). https://doi.org/10.1017/CBO9781139012843.018
20. Galbraith, S.D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. 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 I 23. pp. 3–33. Springer (2017)
21. Kirschmer, M., Voight, J.: Algorithmic enumeration of ideal classes for quaternion orders. SIAM Journal on Computing **39**(5), 1714–1747 (2010)
22. Kohel, D., Lauter, K., Petit, C., Tignol, J.P.: On the quaternion-isogeny path problem. LMS Journal of Computation and Mathematics **17**(A), 418–432 (2014)
23. Lagarias, J.: Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. Journal of Algorithms **1**(2), 142–186 (1980). https://doi.org/https://doi.org/10.1016/0196-6774(80)90021-8, https://www.sciencedirect.com/science/article/pii/0196677480900218
24. Lenstra, A.K., Lenstra, H.W., Lovász, L.M.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**, 515–534 (1982)
25. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: 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. pp. 448–471. Springer (2023)
26. National Institute for Standards and Technology (NIST): Post-quantum crypto standardization (2016), https://csrc.nist.gov/projects/post-quantum-cryptography
27. National Institute for Standards and Technology (NIST): Post-quantum crypto standardization (2016), https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals
28. Peikert, C.: He gives c-sieves on the csidh. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30. pp. 463–492. Springer (2020)
29. Robert, D.: 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. pp. 472–503. Springer (2023)

30. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. IACR Cryptology ePrint Archive **2006**, 145 (2006)
31. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)
32. Silverman, J.H.: The arithmetic of elliptic curves, vol. 106. Springer (2009)
33. Stein, J.: Computational problems associated with racah algebra. Journal of Computational Physics **1**(3), 397–405 (1967). https://doi.org/https://doi.org/10.1016/0021-9991(67)90047-2, https://www.sciencedirect.com/science/article/pii/0021999167900472
34. Vallée, B.: Gauss' algorithm revisited. Journal of Algorithms **12**(4), 556–572 (1991). https://doi.org/https://doi.org/10.1016/0196-6774(91)90033-U, https://www.sciencedirect.com/science/article/pii/019667749190033U
35. Voight, J.: Quaternion algebras. Springer Nature (2021)