CSI-FiSh: Efficient Isogeny based Signatures through Class Group Computations

Ward Beullens[1], Thorsten Kleinjung[2], and Frederik Vercauteren[1]

ward.beullens@esat.kuleuven.be, thorsten.kleinjung@epfl.ch, frederik.vercauteren@esat.kuleuven.be

[1] imec-COSIC, ESAT, KU Leuven, Belgium
[2] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland

Abstract. In this paper we report on a new record class group computation of an imaginary quadratic field having 154-digit discriminant, surpassing the previous record of 130 digits. This class group is central to the CSIDH-512 isogeny based cryptosystem, and knowing the class group structure and relation lattice implies efficient uniform sampling and a canonical representation of its elements. Both operations were impossible before and allow us to instantiate an isogeny based signature scheme first sketched by Stolbunov, which we further optimize using multiple public keys and Merkle trees. We also show that including quadratic twists allows to cut the public key size in half for free. Optimizing for signature size, our implementation takes 390ms to sign/verify and results in signatures of 263 bytes, at the expense of a large public key. This is 300 times faster and over 3 times smaller than an optimized version of SeaSign for the same parameter set. Optimizing for public key and signature size combined, results in a total size of 1468 bytes, which is smaller than any other post-quantum signature scheme at the 128-bit security level.

Keywords: Isogeny-based cryptography, digital signature, class group, group action, Fiat-Shamir.

1 Introduction

Isogeny-based cryptography was first proposed in 1997 by Couveignes [8] in a talk at the séminaire de complexité et cryptographie at the ENS, but his ideas on how class group actions could be used in cryptography were not published at that time. The same idea was independently rediscovered in 2006 by Rostovtsev and Stolbunov [29]. Both Couveignes as well as Rostovtsev and Stolbunov (CRS) described an isogeny based identification scheme and Diffie-Hellman like key agreement scheme using the class group of the endomorphism ring of ordinary elliptic curves; however, neither scheme can be considered practical.

[*] This work was supported in part by the Research Council KU Leuven grants C14/18/067 and STG/17/019. Ward Beullens is funded by an FWO fellowship. Date of this document: 2019.05.14.

A different approach was taken by Jao and De Feo who introduced SIDH (Supersingular Isogeny Diffie–Hellman) [21]. SIDH does not rely on class group actions as done by CRS, but exploits the simple fact that dividing out an elliptic curve by two (large) non-intersecting subgroups is commutative. SIDH uses supersingular curves, mainly for two reasons: firstly, constructing a supersingular elliptic curve with given group order is trivial, and secondly, their endomorphism ring is non-commutative which thwarts attacks by Kuperberg's algorithm [24]. SIDH forms the basis of a practical key-exchange protocol called SIKE [20], which is one of the main contenders in NIST's post-quantum standardization project [28].

A major improvement of CRS was made by Castryck et. al. [5] by instantiating the scheme for supersingular curves over $\mathbb{F}_p$ and by restricting the endomorphism ring to $\mathbb{F}_p$-rational endomorphisms. This subring behaves very much like in the ordinary curve setting, so the CRS approach applies. The main advantage is that the class group action can be computed very efficiently since by construction, the supersingular curves have many small rational subgroups. The resulting cryptosystem is called CSIDH for Commutative Supersingular Isogeny Diffie-Hellman and is pronounced "sea-side".

Both SIDH and CSIDH result in efficient key-agreement schemes, but a practical isogeny-based signature scheme is much harder to achieve. The first attempt was made by Stolbunov in his PhD thesis [33]; the signature scheme consists of the Fiat-Shamir transform applied to a standard three pass isogeny-based identification scheme. The scheme can be securely instantiated under two assumptions: firstly, it should be possible to sample uniformly in the class group (this could be efficiently approximated) and secondly, each element in the class group has an efficiently computable canonical representation. Especially the second assumption is a major obstacle to instantiate Stolbunov's signature scheme.

This problem was partly remedied by De Feo and Galbraith in the signature scheme SeaSign [10] by employing "Fiat–Shamir with aborts". The main idea is, instead of using a canonical representation for each class group element, to use a majorly redundant representation and to apply rejection sampling to make the distribution of the class group elements, which are part of the signature, independent of the secret key. Several versions of SeaSign were presented offering trade-offs between signature size, public-key size, and secret-key size. Although signature sizes of less than one kilobyte at the 128-bit security level are possible, the scheme is again not practical taking several minutes to sign. Decru et al. [11] improved all variants of SeaSign, but the fastest parameter set still requires 2 minutes to sign a message.

A different approach was taken by Yoo et al. [36] who transform an SIDH-based zero-knowledge proof proposed by De Feo et al. [14] into a digital signature scheme. The resulting signatures however are rather large at $\sim$ 120KB which is much larger than other post-quantum signature schemes. A similar approach was described by Galbraith et al. [16] who were able to compress the signatures down to roughly 10KB. None of the above signature schemes is therefore practical, either due to lack of efficiency or due to the large signatures.

It is well known (see for instance Couveignes [8], Stolbunov's PhD [33] or Section 9.2 of [10]), that knowing the class group structure would resolve the two main problems with Stolbunov's signature scheme. Firstly, uniform sampling is now trivial, but more importantly, each element has an efficiently computable canonical representation. This immediately implies that rejection sampling is no longer necessary, thereby majorly speeding up the resulting signature scheme.

The computation of the class group of a quadratic imaginary number field is a classical problem in computational number theory, and the current best algorithms [19, 4, 22] are improvements of an algorithm due to Hafner and McCurley [17]. These algorithms have complexity $L_{1/2}(\Delta)$ with Δ the discriminant of the number field. The largest publicly known class group computation was for a 130-digit discriminant by Kleinjung [22].

The main contributions in this paper are as follows:

- We compute the class group structure and a relation lattice of the class group of the quadratic imaginary field corresponding to the CSIDH-512 parameter set having a 154-digit discriminant. This computation is described in Section 3.
- We present an efficient algorithm to compute the class group action of random class group elements by solving an approximate CVP-problem in the relation lattice. This strategy is described in Section 4 and is a combination of Babai nearest plane algorithm [1] and a random walk approach due to Doulgerakis, Laarhoven and de Weger [13]. Compared to native CSIDH which starts from an efficient representation, our algorithm is only 15% slower.
- In Section 5, we introduce CSI-FiSh (Commutative Supersingular Isogeny based Fiat-Shamir signatures, pronounce "sea-fish") which is based on Stolbunov's signature scheme [33] combined with optimisations similar to the ones described for SeaSign [10]. We also show that the public key size can be cut in half for free by including not only the curve, but also its quadratic twist. This implicitly doubles the number of curves in the public key for free, without affecting the security of the scheme. Finally, we prove that the resulting signature scheme is secure in the quantum random oracle model.
- We provide an efficient open-source implementation of CSI-FiSh and report on the implementation results in Section 6. As for SeaSign, CSI-FiSh allows for various trade-offs: the smallest signatures are 263 bytes and are also the fastest ($\sim$ 390ms to sign/verify), but require a large public key of 2 MB. Slightly larger signatures of 461 bytes require a public key of 16KB which is comparable to multivariate schemes such as LUOV [3], but take $\sim$ 670ms to compute. Optimizing for public key and signature size combined, results in a total size of 1468 bytes which is smaller than any other post-quantum signature scheme at the 128-bit security level.

2 Preliminaries

We denote by $[a,b]$ with $a,b \in \mathbb{Z}, a \leq b$ the set $\{a,\ldots,b\}$. When considering reals instead of integers $[a,b]$ denotes the interval $a \leq r \leq b$ with $r \in \mathbb{R}$, whereas $[a,b[$ denotes $a \leq r < b$. The cardinality of a set S is denoted by $\#S$.

2.1 Elliptic curves and isogenies

The goto general reference on elliptic curves is Silverman [31]. A good introduction to isogeny based cryptography can be found in the lecture notes by De Feo [9].

Let E be an elliptic curve over a finite field $\mathbb{F}_p$ with p a large prime, and let $\mathbf{0}_E$ denote the point at infinity on E. The curve E is called supersingular iff $\#E(\mathbb{F}_p) = p+1$, and ordinary otherwise. Given two elliptic curves E and E', an isogeny ϕ is a morphism $\phi : E \to E'$ (i.e. can be expressed as fractions of polynomials) such that $\phi(\mathbf{0}_E) = \mathbf{0}_{E'}$. An isomorphism is an isogeny whose inverse over the algebraic closure is also an isogeny and two elliptic curves are isomorphic iff they have the same j-invariant, which is a simple algebraic expression in the coefficients of the curve. Since an isogeny defines a group homomorphism from E to E', its kernel is a subgroup of E. Vice-versa, any subgroup $S \subset E(\mathbb{F}_{p^k})$ determines a (separable) isogeny $\phi : E \to E'$ with $\ker \phi = S$, i.e. $E' = E/S$. The equation for E' and the isogeny ϕ can be computed using Vélu's formulae [34] using $O(\#S(k \log p)^2)$ bit-operations. As such, it is only practical to handle fairly small subgroups S defined over small extensions of $\mathbb{F}_p$.

The ring of endomorphisms $\mathrm{End}(E)$ consists of all isogenies from E to itself, and $\mathrm{End}_{\mathbb{F}_p}(E)$ denotes the ring of endomorphisms defined over $\mathbb{F}_p$. For an ordinary curve $E/\mathbb{F}_p$ we have $\mathrm{End}(E) = \mathrm{End}_{\mathbb{F}_p}(E)$, but for a supersingular curve over $\mathbb{F}_p$ we have a strict inclusion $\mathrm{End}_{\mathbb{F}_p}(E) \subsetneq \mathrm{End}(E)$. In particular, it is known that for a supersingular curve over $\mathbb{F}_p$ its full endomorphism ring $\mathrm{End}(E)$ is an order in a quaternion algebra, whereas $\mathrm{End}_{\mathbb{F}_p}(E)$ is only an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$. In the following we will denote this order $\mathcal{O} = \mathrm{End}_{\mathbb{F}_p}(E)$.

The ideal class group of $\mathcal{O}$ is the quotient of the group of fractional invertible ideals in $\mathcal{O}$ by the principal fractional invertible ideals, and will be denoted $\mathrm{Cl}(\mathcal{O})$. Given an $\mathcal{O}$-ideal $\mathfrak{a}$, we can consider the subgroup defined by the intersection of the kernels of the endomorphisms in $\mathfrak{a}$, i.e. $S_{\mathfrak{a}} = \bigcap_{\alpha \in \mathfrak{a}} \ker \alpha$. Since this is a subgroup of E, we can divide out by $S_{\mathfrak{a}}$ and denote the isogenous curve $E/S_{\mathfrak{a}}$ by $\mathfrak{a} \star E$. This isogeny is well-defined and unique up to $\mathbb{F}_p$-isomorphism and the group $\mathrm{Cl}(\mathcal{O})$ acts via the operator $\star$ on the set $\mathcal{E}$ of $\mathbb{F}_p$-isomorphism classes of elliptic curves with $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O}$. One can show that $\mathrm{Cl}(\mathcal{O})$ acts freely and transitively on $\mathcal{E}$, i.e. $\mathcal{E}$ is a principal homogeneous space for $\mathrm{Cl}(\mathcal{O})$.

In what follows we will assume that the class group $\mathrm{Cl}(\mathcal{O})$ is cyclic of order $N = \#\mathrm{Cl}(\mathcal{O})$ generated by the class of an ideal $\mathfrak{g}$. The more general case of non-cyclic class groups is a trivial extension and is not required in the application we consider.

2.2 CSIDH

Castryck et al. [5] proposed an efficient commutative group action $\star$ by crafting supersingular elliptic curves with many small $\mathbb{F}_p$-rational subgroups. Given that $\#E(\mathbb{F}_p) = p+1$ for a supersingular curve, it is immediate that if p is chosen to be of the form $4 \cdot \ell_1 \cdots \ell_n - 1$, with ℓ_i small distinct odd primes, we have $\#E(\mathbb{F}_p) = 4 \cdot \ell_1 \cdots \ell_n$. Such curves therefore have, for each $i \in [1, n]$, an $\mathbb{F}_p$-rational subgroup of order ℓ_i. Since $p = -1 \bmod \ell_i$, we have that in $\mathbb{Q}(\sqrt{-p})$ the rational prime ℓ_i splits as $(\ell_i) = \langle \ell_i, \pi - 1 \rangle \langle \ell_i, \pi + 1 \rangle$, where $\pi = \sqrt{-p}$ represents the $\mathbb{F}_p$-Frobenius endomorphism. Note that the first ideal factor $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$ corresponds to the subgroup of order ℓ_i defined over $\mathbb{F}_p$, and that the action of this ideal can be computed entirely over $\mathbb{F}_p$. Once this subgroup is determined, Vélu's formulae require $O(\ell_i (\log p)^2)$ bit operations. However, for small ℓ_i, finding a generator of this small subgroup requires (at least one) full-size scalar multiplication which dominates the cost of Vélu's formulae.

CSIDH acts by ideals of the form $\prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ where the exponents are chosen uniformly from some interval $[-B, B]$. This can be done by computing sequentially the action of $\mathfrak{l}_i$ exactly e_i times. Since the cost of each such action is dominated by the cost to determine the correct subgroup, we assume that the overall cost of computing such action is mostly determined by the ℓ_1-norm of its exponent vector, i.e. $|e_1| + \cdots + |e_n|$.

The base curve is taken to be $E_0 \colon y^2 = x^3 + x$ over $\mathbb{F}_p$ and instead of using the j-invariant, each isomorphism class of a curve with given endomorphism ring $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O} = \mathbb{Z}[\pi]$ is represented by a single coefficient $A \in \mathbb{F}_p$ defining the curve $E_A \colon y^2 = x^3 + Ax^2 + x$. Denote $\mathcal{A}$ the set of all such coefficients A, then we obtain a class group action $\star : \mathrm{Cl}(\mathcal{O}) \times \mathcal{A} \to \mathcal{A}$ or equivalently, assuming the class group is cyclic of order N, a group action $[] : \mathbb{Z}_N \times \mathcal{A} \to \mathcal{A}$. To simplify notation in the remainder of the paper, we will identify a curve E_A with its isomorphism class represented by the corresponding coefficient A.

Note however that in CSIDH, the order (and structure) of the class group are unknown, so only the action of ideals of the form $\prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ with e_i smallish are computable. This restriction brings up various questions: firstly, given the range of exponent vectors $[-B, B]^n$, do the ideals $\prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ cover the whole class group, and secondly, assuming the exponents are chosen uniformly in $[-B, B]$, is the resulting distribution of $\prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ uniform over $\mathrm{Cl}(\mathcal{O})$. It is clear that knowing the class group structure voids both questions as surjectivity and uniformity become trivial to attain. The only remaining problem then is to efficiently compute the action $[a]$ given a random exponent $a \in \mathbb{Z}_N$ (see Section 4 for an efficient solution).

2.3 Computational problems

The main hardness assumption underlying group actions based on isogenies, is that it is hard to invert the group action:

Definition 1 (Group Action Inverse Problem (GAIP)). *Given a curve E, with* $\mathrm{End}(E) = \mathcal{O}$*, find an ideal $\mathfrak{a} \subset \mathcal{O}$ such that $E = \mathfrak{a} \star E_0$.*

Another advantage of knowing the class group structure and therefore uniform sampling, is that the GAIP is random self-reducible: given a problem instance E, we can shift this over a uniformly random $\mathfrak{b}$ to obtain $E' = \mathfrak{b} \star E$, which is uniformly distributed in $\mathcal{A}$. Given a solution $\mathfrak{c}$ for E', it is easy to see that $\mathfrak{c}\mathfrak{b}^{-1}$ is then a solution to the original problem.

The CSI-FiSh signature scheme relies on the hardness of random instances of a multi-target version of the inversion problem, which is shown to reduce tightly to the normal GAIP by [10] in the case that the class group structure is known.

Definition 2 (Multi-Target Group Action Inverse Problem (MT-GAIP)). *Given k curves $E_1, \ldots, E_k$ with $\mathrm{End}(E_1) = \cdots = \mathrm{End}(E_k) = \mathcal{O}$, find an ideal $\mathfrak{a} \subset \mathcal{O}$ such that $E_i = \mathfrak{a} \star E_j$ for some $i, j \in \{0, \cdots, k\}$ with $i \neq j$.*

The best classical algorithm to solve the GAIP problem is a simple meet-in-the-middle approach, where one finds a collision between two breadth-first trees starting at E and E' respectively. The time complexity of this approach is $O(\sqrt{\#\mathrm{Cl}(\mathcal{O})})$. The best quantum algorithm for the GAIP problem reformulates it as a hidden shift problem [6] and then applies Kuperberg's algorithm [24, 25], which runs in time $2^{O(\sqrt{\log N})}$. Translating this subexponential complexity to concrete security estimates is a highly non-trivial endeavour and we refer to [5, Section 7] for precise details.

In this paper we will only focus on the CSIDH-512 parameter set, which uses 74 small primes ℓ_i (so $n = 74$) and samples the exponents uniformly from the interval $[-5, 5]$ (so $B = 5$). This parameter set is estimated to provide 128-bit classical security and to achieve NIST security level 1 quantumly [5]. The CSIDH authors assume that sampling exponent vectors in $[-5, 5]$ covers a subset of size $\sim 2^{256}$, which is a bit less than half of the total size of the class group. Class group elements (represented by their exponent vectors) require roughly 32 bytes, and each isomorphism class requires 64 bytes (one coefficient in $\mathbb{F}_p$). The average time taken to perform one such group action [5] is roughly 40 ms on a 3.5GHz processor.

3 Class group computation

In order to uniformly sample and canonically represent class group elements, a class group computation of Hafner-McCurley type [17] was performed which, besides computing generators of the class group, also expresses the ideal classes of prime ideals with small norm in terms of these generators. This computation relied on the programs from [22], which work over the maximal order and thus we obtain generators for $\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$. This class group turns out to be cyclic and the class number is not divisible by 3. Since the conductor of the suborder $\mathcal{O}$ is (2) and 2 does not split in $\mathcal{O}_{\mathbb{Q}(\sqrt{-p})}$, we get $\#\mathrm{Cl}(\mathcal{O}) = 3\#\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ so that $\mathrm{Cl}(\mathcal{O})$ is also cyclic. Using the information from the computation over the maximal order, it is easy to find a generator of $\mathrm{Cl}(\mathcal{O})$ and to express the $\mathfrak{l}_i$ as powers of this generator. In total, the computation took an estimated effort of 52 core years on an inhomogenous cluster of number crunchers and desktop machines, consisting of around 800 cores with the "average" core running at around 3.3GHz.

The class group computation consists of the following steps.

Relation collection. Given a bound F (we chose $F = 7000000$), let $\mathcal{F}$ be the set of prime ideals of degree one with norm less than F and the prime ideal (2); the latter is only included for technical reasons. A relation is a decomposition $(a+\sqrt{-p}) = \prod_{\mathfrak{p}\in\mathcal{F}} \mathfrak{p}^{e_{a,\mathfrak{p}}}$ with $a, e_{a,\mathfrak{p}} \in \mathbb{Z}$. Such relations can be found by factoring the ideal $(a + \sqrt{-p})$ for random $a \in \mathbb{Z}$ which essentially amounts to factoring its norm $a^2 + p$. Since most a do not give rise to a relation, there exist many methods to speed up the search for relations. We used a sieving approach [22] and the large prime variation with up to three large primes; these details do not matter in the following and are suppressed.

The goal of this step is to generate sufficiently many relations such that the subsequent steps are able to determine the class group. In practice, this usually means that we can stop collecting relations when the number of relations slightly exceeds the number of prime ideals contained in their decompositions (which is at most $\#\mathcal{F}$). However, a bigger excess often reduces the running time of the subsequent steps significantly.

This step is one of the two main steps in terms of computational effort. Fortunately, it is trivially parallelized and has moderate memory requirements. In our computation it took an estimated time of 43 core years to collect 319.5 million relations over an extended factor base of size 32.7 million.

Building the matrix. In this step the set of relations is converted into a matrix over $\mathbb{Z}$ with rows corresponding to prime ideals and columns corresponding to relations; the matrix entry belonging to the prime ideal $\mathfrak{p} \in \mathcal{F}$ and relation $(a + \sqrt{-p})$ is $e_{a,\mathfrak{p}}$. This matrix is overdetermined and very sparse. We now assume that the ideal classes of the prime ideals in $\mathcal{F}$ generate the class group. In practice, it is very likely that this assumption holds; moreover, it follows from GRH if F is chosen appropriately. Under the assumption above, one has a surjection $\mathbb{Z}^{\#\mathcal{F}}/\Lambda \to \mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ where Λ is the lattice spanned by the columns. If the matrix has full rank, the covolume of Λ is a multiple of the class number. By performing elementary column operations as well as removing certain rows and columns one can reduce this matrix significantly while keeping it slightly overdetermined and sparse; this is done to reduce the complexity of the next steps.

In terms of running time this step is negligible but it has higher memory requirements and is not easily parallelisable. We reduced our set of 319.5 million relations over a factor base of size 32.7 million to a slightly overdetermined matrix with roughly 222 thousand rows.

Matrix step. By dropping some columns from the matrix above one can obtain a square matrix and use the (block) Wiedemann algorithm modulo many small primes to compute its determinant over $\mathbb{Z}$ (cf. [35, 7]). If the determinant is non-zero, it is a (usually) huge multiple of the class number. By repeating the determinant calculation for another square matrix obtained by dropping another set of columns one gets a second huge multiple of the class number. Their greatest common divisor is much smaller, thus can be factored, and for each of its prime factors one can check whether it is a divisor of the class number using quadratic forms.

This is the other main step, it is also easy to parallelize and has moderate memory requirements. For both determinant computations, we computed the determinant modulo roughly 7000 different 64-bit primes, which took roughly 4.3 core years per determinant. By taking the gcd of the determinants and removing an extra factor of 2, we obtained that

$$\begin{aligned}\#\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})}) = 37 \times 1407181 \times 51593604295295867744293584889 \\ \times 31599414504681995853008278745587832204909\,.\end{aligned}$$

The class group of the order $\mathcal{O}$ therefore has cardinality $3 \cdot \#\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ which is approximately equal to $2^{257.136}$.

Final computations. In this step the r-Sylow group of $\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ is computed for each r dividing the class number together with the images of all involved prime ideals in this Sylow group. For small r this is easy and for large r the kernel of one of the square matrices from the previous step can be computed modulo r, e.g., using the Lanczos or Wiedemann algorithm. Finally, tying everything together a set of generators of the class group and for each involved prime ideal a representation in terms of these generators are obtained.

This step is negligible in terms of running time and has only moderate memory requirements. It turns out that the ideal $\mathfrak{l}_1 = \langle 3, \pi - 1 \rangle$ generates $\mathrm{Cl}(\mathcal{O})$, the discrete logs of the other $\mathfrak{l}_i$ are available in our GitHub repository [2].

Remark 3. Notice that all odd primes up to 373 split in $\mathbb{Q}(\sqrt{-p})$ thus improving the probablity that the ideal $(a + \sqrt{-p})$ gives rise to a relation. This facilitates the class group computation for our choice of p but the gain is much less than a factor of 2 compared to an average prime of the size of p.

4 Class group action

To compute the action $[]: \mathbb{Z}_N \times \mathcal{A} \to \mathcal{A}$ efficiently, we need to find an equivalent representation of the ideal class of $\mathfrak{g}^a$ for random $a \in \mathbb{Z}_N$. Recall that for isogenies, there is no analogue of the standard square-and-multiply for exponentiation, so a different approach is required. CSIDH acts efficiently by the ideals $\mathfrak{l}_i$ for $i = 1, \ldots, n$, and we will furthermore assume that these ideals generate the class group $\mathrm{Cl}(\mathcal{O})$ and that we have expressed $\mathfrak{g} = \prod_{i=1}^{n} \mathfrak{l}_i^{g_i}$. In practice, it will often be the case that one of the $\mathfrak{l}_i$ generates the class group already, and in fact, for the CSIDH-512 class group we can even take $\mathfrak{g} = \mathfrak{l}_1 = \langle 3, \pi - 1 \rangle$.

As such, we are given an exponent vector $\mathbf{e} = [e_1, \dots, e_n]$ with possibly large entries, that needs to be reduced modulo the relation lattice:

$$L := \{\mathbf{z} = (z_1, \dots, z_n) \in \mathbb{Z}^n : \prod_{i=1}^{n} \mathfrak{l}_i^{z_i} = (1)\} .$$

The lattice L has rank n and volume $N = \#\mathrm{Cl}(\mathcal{O})$ since by definition it is the kernel of the surjective group homomorphism that maps $\mathbb{Z}^n \to \mathrm{Cl}(\mathcal{O}) : \mathbf{z} = (z_1, \dots, z_n) \mapsto \prod_{i=1}^{n} \mathfrak{l}_i^{z_i}$. Note that the relation lattice follows directly from the class group computation described in Section 3.

Since the complexity of a CSIDH action is mainly determined by the ℓ_1-norm of the exponent vector, we want to solve the Closest Vector Problem (CVP) in L for the ℓ_1-norm given the target vector $\mathbf{e}$. Indeed, any vector $\mathbf{z} \in L$ which is close to $\mathbf{e}$ for the ℓ_1 norm will result in an equivalent vector $\mathbf{e} - \mathbf{z}$ such that $\|\mathbf{e} - \mathbf{z}\|_1$ is small and thus efficiently computable.

A first approximation to solving the CVP for the ℓ_1-norm is to use either Babai's rounding or nearest plane algorithm [1]. Given a set of basis vectors $B := \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$, denote with $B^\star := \{\mathbf{b}_1^\star, \dots, \mathbf{b}_n^\star\}$ the corresponding Gram-Schmidt orthogonalization vectors. Let $\mathcal{P}(B)$ denote the parallelepiped

$$\mathcal{P}(B) = \left\{ \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \mid \alpha_i \in [-1/2, 1/2[\right\} ,$$

then Babai rounding returns a lattice vector in $\mathbf{e} + \mathcal{P}(B)$ and Babai's nearest plane in $\mathbf{e} + \mathcal{P}(B^\star)$. This shows that $\mathbf{e} - \mathbf{z}$ is either in $\mathcal{P}(B)$ or in $\mathcal{P}(B^\star)$ depending on the choice of algorithm. Given a basis B and corresponding Gram-Schmidt basis $B^\star$, it is therefore easy to bound $\|\mathbf{e} - \mathbf{z}\|_1$. This also shows that a basis with short and almost orthogonal vectors will give better results. In our experiments, we only used Babai's nearest plane algorithm since it is superior to Babai rounding.

Several notions of reductions and corresponding algorithms exist such as LLL [27], BKZ [30] or HKZ [23]. Since the lattice L is fixed for a given class group, a considerable effort can be spent in reducing the lattice basis during a precomputation. To analyze the impact of the quality of the basis, we computed three reductions: BKZ-40, BKZ-50 and HKZ. For each reduced basis, we then ran Babai nearest plane resulting in Table 1, where the average ℓ_1-norm and standard deviation are given for a sample size of 10^4 random exponents.

The above table should be compared with the expected ℓ_1-norm and standard deviation of vectors sampled according to the CSIDH distribution, i.e. uniform random in $[-B, B]^n$. For $B = 5$ and $n = 74$, we obtain $\mu = n2(5 + 4 + 3 + 2 + 1)/11 = 201.81$ and $\sigma = 13.76$, but note $(2B + 1)^{74} < N/2.2$ so less than half of the class group is covered by CSIDH.

To lower the ℓ_1-norm further, we can employ an algorithm due to Doulgerakis, Laarhoven and de Weger [13] (originally described in [26]). The idea of this algorithm is pretty simple: given a list $\mathcal{S}$ of short vectors in the lattice L, it tries to construct a vector that is closer than the current vector $\mathbf{z}$ by considering $\mathbf{z} \pm \mathbf{s}$

Table 1. ℓ_1-norm and ℓ_2-norm of Babai's nearest plane method and evaluation times of CSIDH-action on three different bases

	BKZ-40	BKZ-50	HKZ
ℓ_1-norm	$\mu = 240.67$ $\sigma = 18.82$	$\mu = 239.35$ $\sigma = 18.35$	$\mu = 237.50$ $\sigma = 18.26$
ℓ_2-norm	$\mu = 35.13$ $\sigma = 2.47$	$\mu = 34.93$ $\sigma = 2.43$	$\mu = 34.67$ $\sigma = 2.38$
action evaluation time (10^6 cycles)	$\mu = 148.59$ $\sigma = 12.91$	$\mu = 148.41$ $\sigma = 12.57$	$\mu = 147.16$ $\sigma = 12.46$

for all $\mathbf{s} \in \mathcal{S}$. This procedure is then repeated on small random shifts of the target vector. The resulting DLW algorithm is described in Algorithm 1.

Algorithm 1 DLW algorithm - randomized slicer for solving CVP

Input: A list $\mathcal{S} \subset L$ of short vectors, target vector $\mathbf{e} \in \mathbb{Z}^n$, number of iterations M
Output: Approximate closest lattice vector $\mathbf{z}$ to $\mathbf{e}$

```
 1: $\mathbf{z} \leftarrow \mathbf{0}$
 2: for $i = 0, \ldots, M-1$ do
 3:     Randomize $\mathbf{e}$ with random small lattice vector to obtain $\mathbf{e}'$
 4:     for $\mathbf{s} \in \mathcal{S}$ do
 5:         if $\|\mathbf{e}' - \mathbf{s}\|_1 < \|\mathbf{e}'\|_1$ then
 6:             $\mathbf{e}' \leftarrow \mathbf{e}' - \mathbf{s}$ and restart for loop in line (4)
 7:         end if
 8:     end for
 9:     if $\|\mathbf{e}'\|_1 < \|\mathbf{e} - \mathbf{z}\|_1$ then
10:         $\mathbf{z} \leftarrow \mathbf{e} - \mathbf{e}'$
11:     end if
12: end for
13: return $\mathbf{z}$
```

We ran Algorithm 1 for varying sizes of lists of short vectors and varying number of iterations; the results can be found in Table 2.

Our experiments indicate that (on our setup) the fastest approach is to use the Babai nearest plane method with 2 iterations of the DLW algorithm, with a list of 10000 short vectors. In this case, the reduction takes $7.2 \cdot 10^6$ cycles on average, and evaluating the CSIDH action takes on average $128.1 \cdot 10^6$ cycles. In comparison, standard CSIDH-512 uses vectors sampled uniformly from $[-5, 5]^{74}$ (which does not sample uniformly from $\mathrm{Cl}(\mathcal{O})$) and takes on average $117.7 \cdot 10^6$ cycles. Hence, the additional cost of sampling uniformly is only 15%.

Table 2. ℓ_1-norm, ℓ_2-norm and evalutation time (reduction + action) of the DLW algorithm combined with Babai's nearest plane method on an HKZ basis

List size	Iterations	ℓ_1-norm	ℓ_2-norm	time of reduction + action
1000	1	223.54 ± 13.29	34.07 ± 2.45	140.17 ± 10.32
1000	3	221.38 ± 11.82	33.79 ± 2.26	138.02 ± 10.24
1000	10	216.84 ± 10.14	33.21 ± 2.03	137.66 ± 9.82
3000	1	219.02 ± 12.02	33.65 ± 2.34	138.09 ± 10.25
3000	3	214.96 ± 10.33	33.03 ± 2.09	136.78 ± 9.46
3000	10	208.75 ± 8.55	32.12 ± 1.81	136.95 ± 8.73
10000	1	213.96 ± 10.92	33.09 ± 2.30	135.55 ± 9.53
10000	3	207.97 ± 9.10	32.08 ± 1.93	135.41 ± 8.82
10000	10	201.26 ± 7.47	31.05 ± 1.66	144.26 ± 7.94

5 The signature scheme

The basic version of CSI-FiSh was already sketched by Stolbunov in his thesis [33, 2.B]. He applies the Fiat-Shamir transform [15] to an isogeny-based identification scheme by Couveignes [8] and independently by Stolbunov [32].

$$\begin{tikzcd}
& E & \\
E_0 \arrow[ur, "\mathfrak{b}\star"] \arrow[rr, "\mathfrak{a}\star"] & & E_1 \arrow[ul, dashed, bend right, "\mathfrak{r}\star"']
\end{tikzcd}$$

Figure 1. The basic identification scheme for challenge $c = 1$.

5.1 The basic identification scheme

The identification scheme is illustrated in Figure 1 and works as follows: the public key of the prover consists of $E_1 = \mathfrak{a} \star E_0$ with $\mathfrak{a}$ a random element in $\mathrm{Cl}(\mathcal{O})$ and E_0 the base curve specified by the system parameters. Assuming that $\mathrm{Cl}(\mathcal{O})$ is cyclic with generator $\mathfrak{g}$, we can write $\mathfrak{a} = \mathfrak{g}^a$ with a random in $\mathbb{Z}_N$ and $N = \#\mathrm{Cl}(\mathcal{O})$. The prover samples a random element $\mathfrak{b} = \mathfrak{g}^b$ with $b \in_R \mathbb{Z}_N$ and commits to the (isomorphism class of the) curve $E = \mathfrak{g}^b \star E_0 = [b]E_0$. The verifier then chooses a random bit $c \in \{0, 1\}$ and sends this to the prover. If $c = 0$, the prover responds with $r = b$, and the verifier checks that $E = [r]E_0$, if $c = 1$, the prover responds with $r = b - a \bmod N$ and the verifier checks that $E = [r]E_1$. Note that reducing modulo N is required to avoid any leakage on a and that the check can be written as $E = [r]E_c$. A detailed description of the protocol is displayed in Figure 2.

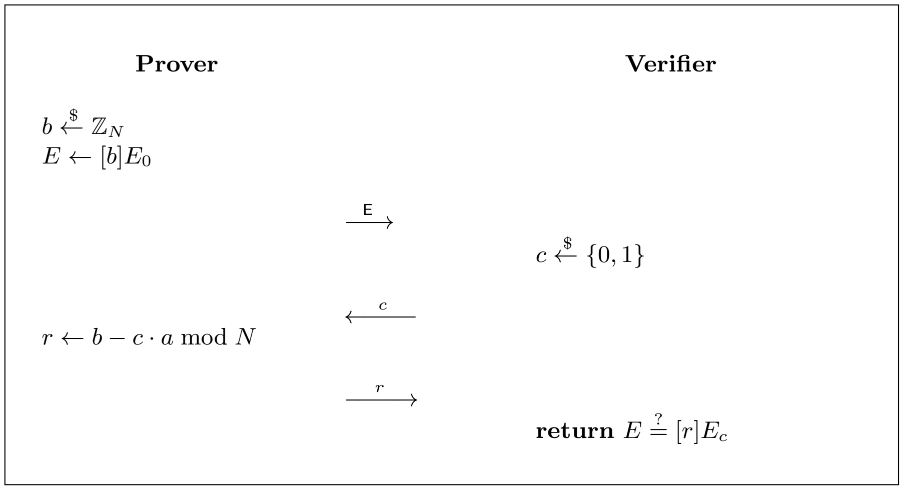
Prover
Verifier
$b \xleftarrow{\$} \mathbb{Z}_N$
$E \leftarrow [b]E_0$
$\xrightarrow{E}$
$c \xleftarrow{\$} \{0,1\}$
$\xleftarrow{c}$
$r \leftarrow b - c \cdot a \bmod N$
$\xrightarrow{r}$
return $E \overset{?}{=} [r]E_c$

Figure 2. The identification scheme of Couveignes and Stolbunov.

Theorem 4. *The Couveignes-Stolbunov protocol (Figure 2) is a complete and secure Sigma protocol proving knowledge of a solution of a GAIP instance. That is, it enjoys completeness, special soundness and special Honest-Verifier Zero Knowledge.*

Proof. **Completeness.** Suppose the protocol is followed honestly, and suppose $E_1 = [a]E_0$. In the case $c = 0$ the verifier checks if $E = [b]E_0$, which is true by construction of E. In the case $c = 1$ the verifier checks if $E = [b-a]E_1$ which holds because

$$[b-a]E_1 = [b-a][a]E_0 = [b]E_0 = E\,.$$

Special Soundness. Suppose $(E, 0, r_0)$ and $(E, 1, r_1)$ are two transcripts that are accepted by the verifier. Then we have

$$E = [r_0]E_0 = [r_1]E_1\,,$$

from which it follows that $[r_0 - r_1]E_0 = E_1$. Hence, it is trivial to extract $r_0 - r_1$, which is a solution to the GAIP problem.

Special Honest-Verifier Zero Knowledge. Consider the simulator that, given a bit c picks a random $r \in \mathbb{Z}_N$, computes $E = [r]E_c$ and outputs the transcript (E, c, r). Then it is clear that the transcripts generated by the simulator are indistinguishable from transcripts of honest executions of the protocol with challenge equal to c: both the real transcripts and the simulated transcripts have uniformly random distributed values of r, and $E = [r]E_c$. □

5.2 Optimizing the Sigma protocol

Hashing. To reduce the communication cost (and hence the signature size after applying the Fiat-Shamir transform) it suffices for the Prover to send $\mathcal{H}(E)$

rather than E, for some collision resistant hash function $\mathcal{H}$. The verifier then computes $\mathcal{H}([r]E_c)$ and checks that it is equal to the hash value sent by the prover. If we are doing t rounds of the protocol in parallel to amplify soundness, it suffices to send a single hash of the concatenation of all the $E^{(i)}$ for i from 1 to t. Clearly the completeness and the Honest-Verifier Zero Knowledge properties of the scheme are not affected by this change. For special soundness, the collision resistance of $\mathcal{H}$ implies that if

$$\mathcal{H}([r_1^{(1)}]E_{c_1^{(1)}}||\cdots||[r_1^{(t)}]E_{c_1^{(t)}}) = \mathcal{H}([r_2^{(1)}]E_{c_2^{(1)}}||\cdots||[r_2^{(t)}]E_{c_2^{(t)}})$$

then $[r_1^{(i)}]E_{c_1^{(i)}} = [r_2^{(i)}]E_{c_2^{(i)}}$ for all i from 1 to t. Hence, if we model $\mathcal{H}$ as a random oracle it is sufficient for $\mathcal{H}$ to have output length 2λ, with λ the security level.

Larger challenge spaces. A well-known approach [10] to lower the soundness error is to increase the challenge space. To do this we move from the GAIP problem to the MT-GAIP problem. We now have $S-1$ public keys instead of one, i.e. the public key now consists of the S-tuple $(E_0, E_1 = [a_1]E_0, \dots, E_{S-1} = [a_{S-1}]E_0)$ (note that E_0 can be left out, it is just there to illustrate the notation) and the prover proves to the verifier that he knows an $s \in \mathbb{Z}_N$ such that $[s]E_i = E_j$ for some pair of curves in the public key (with $i \neq j$). The prover still chooses a random exponent $b \in_R \mathbb{Z}_N$ and computes $E^{(i)} = [b]E_0$. The verifier now sends a challenge $c \in [0, S[$, and the response consists of $r = b - a_c \bmod N$. The verifier then recomputes $[r]E_c$ and verifies that this is equal to $E^{(i)}$. Theorem 4 generalizes to the new identification scheme. In particular, since the challenge space now contains S elements the soundness error drops to $1/S$.

Theorem 5. *The adapted identification scheme is a complete and secure Sigma protocol proving knowledge of a solution of an MT-GAIP instance.*

Proof. The proof is completely analogous to the proof of Theorem 4. $\square$

Doubling the challenge space with twists. To increase the size of the challenge space even further, we exploit the fact that given a curve $E = [a]E_0$, its quadratic twist E^t (which can be computed very efficiently) is $\mathbb{F}_p$-isomorphic to $[-a]E_0$ [5]. Therefore, we can almost double the set of public key curves going from $E_0, E_1, ..., E_{S-1}$ to $E_{-S+1}, \cdots, E_0, \cdots, E_{S-1}$, where $E_{-i} = E_i^t$, without any increase in communication cost. Hence, the soundness error drops to $\frac{1}{2S-1}$. Theorem 5 still applies, but instead of a reduction from a random MT-GAIP instance, we now have a reduction from a random MT-GAIP instance subject to $E_{-i} = E_i^t$ (we call this twisted MT-GAIP). However, there is a simple reduction from this problem to GAIP, which shows this optimization does not affect security.

Theorem 6. *Given an adversary $\mathcal{A}$ that solves a random instance of twisted MT-GAIP in time T and with probability ϵ, there exists an adversary $\mathcal{B}^{\mathcal{A}}$ that* *solves a random instance of MT-GAIP in time $T + O(S)$ with probability at least $\epsilon/2$.*

Proof. We describe the adversary $\mathcal{B}^{\mathcal{A}}$. Suppose $\mathcal{B}$ is given a random MT-GAIP instance $E_1, \cdots, E_k$, then he chooses k random bits $b_1, \cdots, b_k$ and defines curves

$$\tilde{E_i} = \begin{cases} E_i \text{ if } b_i = 0 \\ E_i^t \text{ if } b_i = 1 \end{cases} ,$$

then he sets $\tilde{E_0} = E_0$ and $\tilde{E}_{-i} = \tilde{E_i}^t$ for all i in $\{1, \cdots, k\}$. This is a random twisted MT-GAIP instance that $\mathcal{B}$ then sends to $\mathcal{A}$. With probability ϵ, $\mathcal{A}$ responds with (a, i, j) such that $i \neq j$ and $\tilde{E_i} = [a]\tilde{E_j}$. Now we consider 2 cases:

- $i = -j$. In this case we have $\tilde{E_i} = [a]\tilde{E_i}^t$, which implies $\tilde{E_i} = [a/2]E_0$, so $\mathcal{B}$ outputs $((-1)^{b_{|i|}} a/2, |i|, 0)$, which is a valid solution to his MT-GAIP instance ($|\mathrm{Cl}(\mathcal{O})|$ is known to be odd, so the inverse of 2 always exists).
- $|i| \neq |j|$. In this case we have $\mathrm{sign}(i)(-1)^{b_{|i|}} = \mathrm{sign}(j)(-1)^{b_{|j|}}$ with probability $\frac{1}{2}$. In this case we have an equation of the form $E_i = [\pm a]E_j$ or $E_i^t = [\pm a]E_j^t$. Therefore B can output a valid solution to his MT-GAIP problem $(\pm a, |i|, |j|)$.

Shorter public keys. The previous section explains how one can improve the communication cost and the proving and verification time by considering multiple public key curves $E_i = [a_i]E_0$ for $i \in \{1, \cdots, S-1\}$. The drawback of this approach is that the public key now consists of $S-1$ curves, so its size blows up as S increases. Note that at most t of these public key curves are used during each verification (where t is the number of parallel executions of the protocol to amplify soundness). Therefore, instead of including all the curves $E_1, \cdots, E_{S-1}$ in the public key, the public key can just be a commitment to those curves. The improvement in total communication cost comes from the fact that the response of the prover now only has to include the opening of at most t curves $E_{c_1}, \cdots, E_{c_t}$. If the commitment scheme is binding, then a cheating prover cannot open the commitment to an incorrect curve, so the security of the scheme is preserved. We use a Merkle tree construction to implement the binding commitments, because this allows for the efficient opening of a subset of the curves.

In particular, suppose for simplicity that $S - 1 = 2^d$ and let

$$h_{d,i} = \mathcal{H}(E_i || 2^d + i || \mathsf{MerkleKey}) ,$$

where $\mathsf{MerkleKey} \in \{0,1\}^\lambda$ is a key which is chosen uniformly at random during key generation and included in both the secret and public keys. Then we define each internal node of the Merkle tree as the hash of its children, concatenated with its position in the tree and the $\mathsf{MerkleKey}$:

$$h_{k,i} = \mathcal{H}(h_{k+1,2i-1} || h_{k+1,2i} || 2^k + i || \mathsf{MerkleKey}) .$$

It is an easy exercise to show that if we model $\mathcal{H}$ as a random oracle, the root of the Merkle tree is a binding commitment: An adversary making q queries to the random oracle has at most probability $\frac{q+1}{2^\lambda}$ of breaking the binding property. Note that the $\mathsf{MerkleKey}$ is not strictly required to prove soundness, but it prevents an adversary from attacking multiple public keys at the same time. A similar approach of reducing the public key size was proposed by [10]. They use the more complicated and slightly less efficient construction of [18], which is designed to be provably secure in the standard model. Since the Fiat-Shamir transform relies on the (Q)ROM anyway, there is no reason to use this approach.

5.3 Signatures

The above identification schemes can be turned into (non-interactive) signature schemes using the Fiat-Shamir transform [15], where the challenges $c_i \in \{-S+1, \cdots, S-1\}$ are simply obtained by hashing the ephemeral keys $E^{(i)}$ for $i = 1, \dots, t$ together with the message m, i.e. $(c_1, \dots, c_t) = \mathcal{H}(E^{(1)} || \dots || E^{(t)} || m)$. The signature then consists of $(r_1, \dots, r_t, c_1, \dots, c_t)$, and the verifier recomputes the $E^{(i)} = [r_i]E_{c_i}$ and checks that indeed $(c_1, \dots, c_t) = \mathcal{H}(E^{(1)} || \dots || E^{(t)} || m)$. Figure 3 details the "simple" variant and corresponds to the identification scheme using multiple public keys. The "Merkle" variant reduces the size of the public key by using a Merkle tree as described above.

To achieve security level λ, we require $t = \lambda / \log_2 S$ and the resulting signature size is $t(\lceil \log_2 N \rceil + \lceil \log_2 S \rceil)$ bits, and both signing and verification require t CSIDH actions (including the time to construct a small representant of the ideal).

The results on Fiat-Shamir in the QROM of Don et al. [12] readily apply to our setting:

Theorem 7. *Assume the hash functions used are modeled as quantum random oracles, then CSI-FiSh is sEUF-CMA secure.*

Proof. The basic sigma protocol (without hashing) has special soundness and unique responses (for each i there exists only one value of $r_i \in \mathbb{Z}_N$ such that $[r_i]E_{c_i} = E^{(i)}$). Hence, Theorem 25 of [12] implies that the scheme also has the Quantum Proof of Knowledge property. The protocol also has more than λ bits of min entropy and perfect HVZK, so Theorem 22 of [12] implies that the Fiat-Shamir scheme is sEUF-CMA secure in the QROM.

For the variant with hashing, it is known that Quantum random oracles are collapsing, so it is immediate that the sigma protocol has quantum computationally unique responses. Hence, the claim again follows from Theorems 25 and 22 of [12].

6 Implementation results

6.1 Parameter Choices

Slow Hash functions Because the QROM security proof is very non-tight it would not be practical to choose parameters in such a way that security is

Algorithm 2 KeyGen

```
Input: $E_0$, class number $N = \#\mathrm{Cl}(\mathcal{O})$
Output: $\mathbf{sk}, \mathbf{pk}$
 1: for $i \in \{1, \cdots, S-1\}$ do
 2:     $a_i \leftarrow_R \mathbb{Z}_N$
 3:     $E_i = [a_i]E_0$
 4: end for
 5: $\mathbf{pk} = [E_i : i \in \{1, \cdots, S-1\}]$
 6: return $(\mathbf{sk} = \mathbf{a}, \mathbf{pk})$
```

Algorithm 3 Sign

```
Input: msg, $\mathbf{sk} = \mathbf{a}$
Output: $\sigma = (r_1, \ldots, r_t, c_1, \ldots, c_t)$
 1: $a_0 \leftarrow 0$
 2: for $i = 1, \ldots, t$ do
 3:     $b_i \leftarrow_R \mathbb{Z}_N$, $E^{(i)} = [b_i]E_0$
 4: end for
 5: $(c_1, \ldots, c_t) = \mathcal{H}(E^{(1)}||\ldots||E^{(t)}||m)$
 6: for $i = 1, \ldots, t$ do
 7:     $r_i = b_i - \mathrm{sign}(c_i)a_{|c_i|} \bmod N$
 8: end for
 9: return $\sigma = (r_1, \ldots, r_t, c_1, \ldots, c_t)$
```

Algorithm 4 Verify

```
Input: msg, $E_0, \mathbf{pk} = [E_i : i \in \{1, \cdots, S-1\}], \sigma$
Output: Valid / invalid
 1: Parse $\sigma$ as $(r_1, \ldots, r_t, c_1, \ldots, c_t)$
 2: Define $E_{-i} = E_i^t$ for all $i \in \{1, \cdots, S-1\}$.
 3: for $i = 1, \ldots, t$ do
 4:     $E^{(i)} = [r_i]E_{c_i}$
 5: end for
 6: $(c'_1, \ldots, c'_t) = \mathcal{H}(E^{(1)}||\ldots||E^{(t)}||m)$
 7: if $(c_1, \ldots, c_t) == (c'_1, \ldots, c'_t)$ then
 8:     return Valid
 9: else
10:     return Invalid
11: end if
```

Figure 3. The “simple” variant of the CSI-FiSh signature scheme.

guaranteed by the proof. Instead, as is customary, we assume that the probablity of a successful attack is at most $Q \times E$, where Q is the number of hash function evaluations that an attacker makes, and E is the soundness error of the zero knowledge proof. So usually one would choose the parameters S and t such that $S^{-t} \leq 2^{-\lambda}$. In our implementation we choose a hash function that is a factor 2^k slower than a standard hash function (e.g. SHA-3), therefore it suffices to take our parameters such that $S^{-t} \leq 2^{-\lambda+k}$. We pick k in such a way that the time spent evaluating the slow hash function is small compared to the total signing and verification time. Since we can take smaller parameters this optimization slightly reduces both the signature size and the signing and verification time.

Proposed parameter sets We have implemented several parameter sets for both the "simple" variant and the "Merkle" variant. For the simple variant the secret key is always small and the variable S controls a trade-off between on the one hand small public keys and fast key generation (when S is small), and on the other hand small signatures and fast signing and verification (when S is large). When we use the "Merkle" variant the public key is always small, but the secret key size increases with increasing value of S, because we store the entire Merkle tree to avoid having to recompute the public keys during signing.

Table 3. Parameter choices and benchmark results for the "simple" variant of CSI-FiSh .

S	t	k	$\lvert\mathbf{sk}\rvert$	$\lvert\mathbf{pk}\rvert$	$\lvert\mathbf{sig}\rvert$	KeyGen	Sign	Verify
2^1	56	16	16 B	128 B	1880 B	100 ms	2.92 s	2.92 s
2^2	38	14	16 B	256 B	1286 B	200 ms	1.98 s	1.97 s
2^3	28	16	16 B	512 B	956 B	400 ms	1.48 s	1.48 s
2^4	23	13	16 B	1 KB	791 B	810 ms	1.20 s	1.19 s
2^6	16	16	16 B	4 KB	560 B	3.3 s	862 ms	859 ms
2^8	13	11	16 B	16 KB	461 B	13 s	671 ms	670 ms
2^{10}	11	7	16 B	64 KB	395 B	52 s	569 ms	567 ms
2^{12}	9	11	16 B	256 KB	329 B	3.5 m	471 ms	469 ms
2^{15}	7	16	16 B	2 MB	263 B	28 m	395 ms	393 ms

Table 4. Parameter choices and benchmark results for the "Merkle" variant of CSI-FiSh .

S	t	k	$\lvert\mathbf{sk}\rvert$	$\lvert\mathbf{pk}\rvert$	$\lvert\mathbf{sig}\rvert$	KeyGen	Sign	Verify
2^8	13	11	8 KB	32 B	1995 B	13 s	671 ms	371 ms
2^{10}	11	7	32 KB	32 B	2086 B	52 s	567 ms	567 ms
2^{12}	9	11	128 KB	32 B	2022 B	3.5 m	467 ms	467 ms
2^{15}	7	16	1 MB	32 B	1953 B	28 m	399 ms	402 ms
2^{18}	6	14	8 MB	32 B	1990 B	3.8 h	335 ms	326 ms

6.2 Implementation details and Benchmarking results

Our proof-of-concept implementation is available on GitHub [2]. To evaluate the CSIDH action, we use the `20180826` version of the proof-of-concept implementation by Castryck et al. [5]. Our implementation depends on the eXtended Keccak Code Package for the implementation of SHAKE256, which we have used as hash function, commitment scheme and to expand randomness. The implementation of the Babai nearest plane step depends on the GMP library for its high precision arithmetic.

All our benchmarking experiments are performed on a Dell OptiPlex 3050 machine with Intel Core i5-7500T CPU @ 2.70GHz. The benchmarking results are displayed in Tables 3 and 4.

Remark 8. Like most discrete logarithm based signature schemes, it is possible to precompute the ephemeral keys in CSI-FiSh , i.e. all CSIDH actions can be computed offline, and the online phase then only consists of t modular subtractions, which are extremely fast.

7 Conclusions and open problems

We computed the class group of the imaginary quadratic field that is at the heart of the CSIDH-512 cryptosystem, and exploited the knowledge of the relation lattice to instantiate the first efficient isogeny based signature scheme called CSI-FiSh. The scheme is flexible in that it allows trade-offs between signature sizes, key sizes and the time to sign/verify. One parameter set of CSI-FiSh gives the smallest combined size of public key and signature, compared to any other existing post-quantum secure signature scheme at the 128-bit security level.

The main open problem, given that the class group is cyclic of order N, is to devise an identification scheme where the challenge is taken from $\mathbb{Z}_N$, instead of binary or from the small set $]-S, S[$. Note that the prover can simply mimick the discrete logarithm based constructions since he can now work in the *ring* $\mathbb{Z}_N$, and thus can create the typical response expressing a combination of the ephemeral key, secret key and challenge. The major problem however is how the verifier can verify this combination to be correct, since the group action still only allows to add a known constant in $\mathbb{Z}_N$. The impact of such an identification scheme would be major: the signature size could possibly be as small as 64 bytes, the public key also 64 bytes and signing would require only one CSIDH action taking around 40ms.

Acknowledgements

We would like to thank the department of Electrical Engineering at KU Leuven for providing the necessary computing power through the HTCondor framework. Many thanks also to Léo Ducas for computing the HKZ basis of the relation lattice and discussions on CVP solvers.

References

[1] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[2] Ward Beullens. CSI-FiSh: github repository available at https://github.com/KULeuven-COSIC/CSI-FiSh, 2019.

[3] Ward Beullens and Bart Preneel. Field lifting for smaller UOV public keys. In *International Conference on Cryptology in India*, pages 227–246. Springer, 2017.

[4] Jean-François Biasse. Improvements in the computation of ideal class groups of imaginary quadratic number fields. *Adv. in Math. of Comm.*, 4(2):141–154, 2010.

[5] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An Efficient Post-Quantum Commutative Group Action. In *ASIACRYPT*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018. https://ia.cr/2018/383.

[6] Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *J. Mathematical Cryptology*, 8(1):1–29, 2014.

[7] Don Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Mathematics of Computation*, 62:333–350, 1994.

[8] Jean Marc Couveignes. Hard Homogeneous Spaces., 1997. IACR Cryptology ePrint Archive 2006/291, https://ia.cr/2006/291.

[9] Luca De Feo. Mathematics of isogeny based cryptography, 2017. https://defeo.lu/ema2017/poly.pdf.

[10] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions, 2018. IACR Cryptology ePrint Archive 2018/824. https://ia.cr/2018/824.

[11] Thomas Decru, Lorenz Panny, and Frederik Vercauteren. Faster SeaSign signatures through improved rejection sampling, 2019.

[12] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. *arXiv preprint arXiv:1902.07556*, 2019.

[13] Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Finding closest lattice vectors using approximate Voronoi cells. *PQCRYPTO. Springer*, 2019.

[14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Mathematical Cryptology*, 8(3):209–247, 2014.

[15] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[16] Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification Protocols and Signature Schemes Based on Supersingular Isogeny Problems. In *Advances in Cryptology - ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.

[17] James L. Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American Mathematical Society*, 2:837–850, 1989.

[18] Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In *Public-Key Cryptography–PKC 2016*, pages 387–416. Springer, 2016.

[19] Michael J. Jacobson. Applying sieving to the computation of quadratic class groups. *Math. Comp*, 68:859–867, 1999.
[20] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. SIKE. Submission to [28]. `http://sike.org`.
[21] David Jao and Luca De Feo. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011. `https://ia.cr/2011/506`.
[22] Thorsten Kleinjung. Quadratic sieving. *Mathematics of Computation*, 85(300):1861–1873, 2016.
[23] Aleksandr Korkine and G Zolotareff. Sur les formes quadratiques. *Mathematische Annalen*, 6(3):366–389, 1873.
[24] Greg Kuperberg. A Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. *SIAM J. Comput.*, 35(1):170–188, 2005.
[25] Greg Kuperberg. Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. In *TQC*, volume 22 of *LIPIcs*, pages 20–34. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
[26] Thijs Laarhoven. Sieving for closest lattice vectors (with preprocessing). In *Selected Areas in Cryptography - SAC 2016*, volume 10532 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2017.
[27] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
[28] National Institute of Standards and Technology. Post-Quantum Cryptography Standardization, December 2016. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization`.
[29] Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based on Isogenies, 2006. IACR Cryptology ePrint Archive 2006/145. `https://ia.cr/2006/145`.
[30] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987.
[31] Joseph H Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer, Dordrecht, 2009.
[32] Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Adv. in Math. of Comm.*, 4(2):215–235, 2010.
[33] Anton Stolbunov. Cryptographic schemes based on isogenies, Doctoral thesis, NTNU, 2012.
[34] J. Vélu. Isogénies entre courbes elliptiques. *Comptes Rendus de l'Académie des Sciences de Paris*, 273:238–241, 1971.
[35] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Information Theory*, 32(1):54–62, 1986.
[36] Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A Post-quantum Digital Signature Scheme Based on Supersingular Isogenies. In *Financial Cryptography and Data Security - FC 2017*, volume 10322 of *Lecture Notes in Computer Science*, pages 163–181. Springer, 2017.