# SeaSign: Compact isogeny signatures from class group actions

Luca De Feo and Steven D. Galbraith

[1] Mathematics Department, University of Auckland, NZ. s.galbraith@auckland.ac.nz
[2] Université de Versailles – Saint-Quentin, Paris, France. luca.de-feo@uvsq.fr

**Abstract.** We give a new signature scheme for isogenies that combines the class group actions of CSIDH with the notion of Fiat-Shamir with aborts. Our techniques allow to have signatures of size less than one kilobyte at the 128-bit security level, even with tight security reduction (to a non-standard problem) in the quantum random oracle model. Hence our signatures are potentially shorter than lattice signatures, but signing and verification are currently very expensive.

## 1 Introduction

Stolbunov [38] was the first to sketch a signature scheme based on isogeny problems. Stolbunov's scheme is in the framework of class group actions. However the scheme was not analysed in the post-quantum setting, and a naive implementation would leak the private key. Due to renewed interest in class group actions, especially CSIDH [10] (due to Castryck, Lange, Martindale, Panny and Renes) and the scheme by De Feo, Kieffer and Smith [19], it is of interest to develop a secure signature scheme in this setting. Our main contribution is to use Lyubashevsky's "Fiat-Shamir with aborts" strategy to obtain a secure signature scheme. We also describe some methods to obtain much shorter signatures than in Stolbunov's original proposal.

Currently it is a major problem to get practical signatures from isogeny problems. Yoo et al (see Table 1 of [42]) state signatures of over 100 kilobytes. This can be reduced using some optimisations. For example [22] state approximately 12 kilobytes for their signature scheme (for classical 128-bit security level). In contrast, in this paper we are able to get signatures smaller than a kilobyte, which is better even than lattice signatures. Unfortunately, signing and verification are very slow, but we might just about be able to live with that in certain applications.

We now briefly summarise the main findings in the paper (for more details see Table 2). For the parameters $(n, B) = (74, 5)$ as used in CSIDH [10] we propose a signature scheme whose public key is around 4Mb, signature size is 944 bytes, and verification time is around 8 minutes (signing time is up to three times longer than this, since rejection sampling requires repeating the signing algorithm). For the same parameters we show that one can reduce the public key size to only 32 bytes, but this increases the signature size to around 3kb and does not add any significant additional cost to signing or verification time. One can obtain even shorter signatures by taking different choices of parameters, for example taking $(n, B) = 20, 3275)$ leads to signatures as small as 416 bytes, but we do not have an estimate of the verification time for these parameters.

The paper is organised as follows. Section 3 gives the basic signature scheme concept, that was proposed by Stolbunov, and our secure variant based on Fiat-Shamir with aborts. Section 4 explains how to get shorter signatures, at the expense of public key size, by using challenges that are more than just a single bit. Section 5 shows how to retain the benefit of shorter signatures, while also having a short public key, by using modified Merkle trees. In Appendix B we show how to use our scheme in the context of lossy keys, from which we obtain tight security in the quantum random oracle model via the results of Kiltz, Lyubashevsky and Schaffner [26] (and this security enhancement involves no increase in signature size, though the primes are larger so computations will be somewhat slower).

The name "SeaSign" is a reference to the name CSIDH, which is pronounced "sea-side".

## 2 Background and notation

We use the following notation: $\#X$ is the number of elements in a set $X$; $\log$ denotes the logarithm in base 2; for $B \in \mathbb{N}$ we denote by $[-B, B]$ the set of integers $u$ with $-B \leq u \leq B$.

## 2.1 Elliptic curves, isogenies, ideal class groups

References for elliptic curves over finite fields and isogenies are Silverman [37], Washington [41], Galbraith [20], Sutherland [39] and De Feo [17]. A good reference for ideal class groups and class group actions is Cox [15].

Let $E$ be an elliptic curve over a field $K$ and let $P \in E(K)$ be a point of order $m$. Then there is a unique (up to isomorphism) elliptic curve $E'$ and separable isogeny $\phi : E \to E'$ such that $\ker(\phi) = \langle P \rangle$. Vélu [40] gives an algorithm to compute and equation for $E'$ and rational functions that enable to compute $\phi$. The complexity of this algorithm is linear in $m$ and requires field operations in $K$, so when $K$ is a finite field it has cost $O(m \log(\#K)^2)$ bit operations using standard arithmetic. In the worst case (i.e., when $m$ is large) this algorithm is exponential-time. In practice this computation is only feasible when $m$ is relatively small (say $m < 1000$) and when the field $K$ over which $P$ is defined is not too large (say, at most a few thousand bits).

For an elliptic curve $E$ over a field $K$ we define $\mathrm{End}(E)$ to be the the ring of endomorphisms of $E$ defined over the algebraic closure of $K$, and $\mathrm{End}_K(E)$ to be the the ring of endomorphisms defined over $K$. Since we are mostly concerned with the CSIDH [10] approach, we will be interested in supersingular elliptic curves $E$ such that $j(E) \in \mathbb{F}_p$, where $p$ is a large prime. In this case $\mathrm{End}(E)$ is a maximal order in a quaternion algebra, while $\mathrm{End}_{\mathbb{F}_p}(E)$ is an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$. Indeed, $\mathbb{Z}[\sqrt{-p}] \subseteq \mathrm{End}_{\mathbb{F}_p}(E)$.

We will be concerned with the ideal class group of the order $\mathcal{O} = \mathrm{End}_{\mathbb{F}_p}(E)$. This is the quotient of the group of fractional invertible ideals in $\mathcal{O}$ by the subgroup of principal fractional invertible ideals. Given two invertible $\mathcal{O}$-ideals $\mathfrak{a}, \mathfrak{b}$ we write $\mathfrak{a} \equiv \mathfrak{b}$ if $\mathfrak{a}$ and $\mathfrak{b}$ are equivalent (meaning that $\mathfrak{a}\mathfrak{b}^{-1}$ is a principal fractional $\mathcal{O}$-ideal).

## 2.2 Class group actions and computational problems

Let $p$ be a prime. Let $E$ be an ordinary elliptic curve over $\mathbb{F}_p$ with $\mathrm{End}(E) \cong \mathcal{O}$ or $E$ a supersingular curve over $\mathbb{F}_p$ with $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathcal{O}$ where $\mathcal{O}$ is an order in an imaginary quadratic field. Let $\mathrm{Cl}(\mathcal{O})$ be the ideal class group of $\mathcal{O}$. One can define the action of an $\mathcal{O}$-ideal $\mathfrak{a}$ on the curve $E$ as the image curve $E'$ under the isogeny $\phi : E \to E'$ whose kernel is equal to the subgroup $E[\mathfrak{a}] = \{P \in E(\overline{\mathbb{F}}_p) : \alpha(P) = 0\ \forall \alpha \in \mathfrak{a}\}$. We denote $E'$ by $\mathfrak{a} * E$.

The set $\{j(E)\}$ of isomorphism classes of elliptic curves with $\mathrm{End}(E) \cong \mathcal{O}$ is a principal homogenous space for $\mathrm{Cl}(\mathcal{O})$. Good references for the details are Couveignes [14] and Stolbunov [38]. The key exchange protocol proposed by Couveignes and Stolbunov is for Alice to send $\mathfrak{a} * E$ to Bob and Bob to send $\mathfrak{b} * E$ to Alice; the shared key is $(\mathfrak{a}\mathfrak{b}) * E$.

The difficulty is that if $\mathfrak{a} \subset \mathcal{O}$ is an arbitrary ideal then the subgroup $E[\mathfrak{a}]$ is typically defined over a very large field extension and the computation of $\mathfrak{a} * E$ has exponential complexity. For efficient computation it is necessary to work with ideals that are a product of powers of small prime ideals, so it is necessary to find a "smooth" ideal in the ideal class of $\mathfrak{a}$. Techniques for smoothing an ideal class in the context of isogeny computation were first proposed in [21] and developed further in [9,25,6]. The state of the art is [6] which gives computation $\mathfrak{a} * E$ for any ideal class in subexponential complexity in $\log(\#\,\mathrm{Cl}(\mathcal{O}))$.

Since subexponential complexity is not good enough, for cryptographic applications it is necessary to choose ideals deliberately of the form $\mathfrak{a} = \prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ where $\mathfrak{l}_1, \dots, \mathfrak{l}_n$ are prime $\mathcal{O}$-ideals of small norm $\ell_i$ and where $(e_1, \dots, e_n)$ is an appropriately chosen vector of exponents. Then, the action of $\mathfrak{a}$ can be computed as a composition of isogenies of degree $\ell_i$. Throughout the paper we assume that $\{\mathfrak{l}_1, \dots, \mathfrak{l}_n\}$ is a set of non-principal prime ideals in $\mathcal{O}$, generating $\mathrm{Cl}(\mathcal{O})$, of norm polynomial in the size of the class group. Theoretically we have the bounds $\#\,\mathrm{Cl}(\mathcal{O}) = O(\sqrt{p}\log(p))$ and, assuming a generalised Riemann hypothesis, $\ell_i = O(\log(p)^2)$. In practice one usually takes $\ell_i = O(\log(p))$ for efficiency reasons; heuristically, this is more than enough to generate the class group.

The basic computational assumption is the ideal action problem. Stolbunov called it "Group Action Inverse Problem (GAIP)". The CSIDH paper speaks of hard homogenous spaces and calls the below problem "Key recovery".

*Problem 1.* Given two elliptic curves $E$ and $E_A$ over the same field with $\mathrm{End}(E) = \mathrm{End}(E_A) = \mathcal{O}$. Find an ideal $\mathfrak{a}$ such that $j(E_A) = j(\mathfrak{a} * E)$.

The best classical algorithms for this problem in the general case have exponential time (at least $\sqrt{\#\,\mathrm{Cl}(\mathcal{O})}$ isogeny computations). Childs, Jao and Soukharev [11] were the first to point out that this problem can be formulated as a "hidden shift" problem, and so quantum algorithms for the hidden shift problem can be applied. Hence, there are subexponential-time quantum algorithms for Problem 1 based on the quantum algorithms of Kuperberg [28] and Regev [34]. It is still an active area of research to assess the exact quantum hardness of these problem; see the recent papers by Biasse-Iezzi-Jacobson [7], Bonnetain-Schrottenloher [8], Jao-LeGrow-Leonardi-Ruiz-Lopez (MathCrypt 2018)). But at the very least, Kuperberg's algorithm requires at least $\tilde{O}(2^{\sqrt{\log(p)/2}})$ quantum gates, thus taking

$$p > 2^{2\lambda^2}, \tag{1}$$

where $\lambda$ is the security parameter, should be sufficient to make Problem 1 hard for a quantum computer.

If the ideals $\mathfrak{a}$ in Problem 1 are sampled uniformly at random then the problem admits a random self-reduction: given an instance $(E, E_A)$ one can choose random ideal classes $\mathfrak{b}_1, \mathfrak{b}_2$ and construct the instance $(E_1, E_2) = (\mathfrak{b}_1 * E, \mathfrak{b}_2 * E_A)$, which is now uniformly distributed across the set of pairs of isomorphism classes of curves in the isogeny class. If $\mathfrak{a}'$ is a solution to the instance $(E_1, E_2)$ then any ideal equivalent to the fractional ideal $\mathfrak{a}'\mathfrak{b}_1\mathfrak{b}_2^{-1}$ is a solution to the original instance. This is a nice feature for security proofs that is not shared by SIDH; we use this idea in Section 4.2.

As already mentioned, when instantiating the group action in practice, one must choose parameters that make evaluating isogenies of degree $\ell_i$ as efficient as possible. This is done both by chooseing the $\ell_i$ to have as small norm as possible, and also by arranging that the kernel subgroups $E[\ell_i]$ are defined over as small a field as possible (so that Vélu's formulas can be used). In the ordinary case, De Feo, Kieffer and Smith [19] introduced a method to make the system as efficient as possible (namely, using Vélu's formulas [40] for some primes $\ell_i$, but they were unable to use Vélu's formulas for all the primes they needed). By using supersingular curves over a field $\mathbb{F}_p$ with $p+1 = 4\prod_{i=1}^n \ell_i$, CSIDH [10] manages to apply Vélu's formulas to all primes $\ell_i$. For key exchange, CSIDH samples the exponent vectors $\mathbf{e} = (e_1, \ldots, e_n) \in [-B, B]^n \subseteq \mathbb{Z}^n$ for a suitable constant $B$.

This leads to a special case of Problem 1 where the ideals may not be uniformly distributed in the ideal class group. For further discussion see Definition 1 and the discussion that follows it. In this special case one can also consider a straightforward meet-in-the-middle attack: Let $E$ and $\mathfrak{a} * E$ be given, where $\mathfrak{a} = \prod_{i=1}^n \mathfrak{l}_i^{e_i}$ over $e_i \in [-B, B]$. We compute lists (assume $n$ is even)

$$L_1 = \left\{ \left( \prod_{i=1}^{n/2} \mathfrak{l}_i^{e_i} \right) * E : e_i \in [-B, B] \right\} \quad \text{and} \quad L_2 = \left\{ \left( \prod_{i=n/2+1}^{n} \mathfrak{l}_i^{e_i} \right) * E_A : e_i \in [-B, B] \right\}.$$

If $L_1 \cap L_2 \neq \varnothing$ then we have solved the isogeny problem. This attack is faster than general methods when the set of ideal classes generated is a small subset of $\mathrm{Cl}(\mathcal{O})$. Hence for security we may require

$$(2B+1)^n > 2^{2\lambda}, \tag{2}$$

where $\lambda$ is the security parameter. Further, there is a quantum algorithm due to Tani, which is straightforward to adapt to this problem (we refer to Section 5.2 of De Feo, Jao and Plût [18] for details). This means we might need to take $(2B+1)^n > 2^{3\lambda}$ to have post-quantum security. However, Adj et al. [2] have given an in-depth analysis of these algorithms, and they argue that the naive analysis is unrealistic as it ignores storage costs and other overheads. Instead, they claim the fastest algorithm in practice (even considering quantum adversaries) for this problem is based on an approach by van Oorschot and Wiener that takes time $(2B+1)^n/M^{3/2}$, where $M$ is the number of processors. Since $M < (2B+1)^{n/3}$ in practice, this cost is larger than $(2B+1)^{n/2}$.

Choosing the best values of $B, n, p$ for large choices of $\lambda$ (e.g., satisfying the constraints of equations (1) and (2)) is non-trivial, but will generally lead to sampling in a very small subset of the whole ideal class group.

We remark that Kuperberg's algorithm uses the entire class group, and there seems to be no way to improve the algorithm for the case where the "hidden shift" is sampled from a distribution far from the uniform distribution. We leave the study of this question to future work.

By taking into account the best known attacks, the CSIDH authors propose parameters for the three NIST categories [32], as summarized in Table 1. Note that in all CSIDH instances the set of sampled ideal classes is (heuristically) likely to cover the whole class group. Their implementation of the smallest parameter size CSIDH-1 computes one class group action in under 100ms on a 3.5GHz processor.

| | $n$ | $\log_2 p$ | $B$ | NIST level | classical security | quantum security | message size | private key size |
|---|---|---|---|---|---|---|---|---|
| CSIDH-1 | 74 | 500 | 5 | 1 | 125 bits | 61 qbits | 63B | 32B |
| CSIDH-3 | 131 | 1020 | 7 | 3 | 255 bits | 93 qbits | 128B | 64B |
| CSIDH-5 | 208 | 1787 | 10 | 5 | 447 bits | 129 qbits | 224B | 115B |

**Table 1.** Proposed parameters for CSIDH [10]. Effective parameters $p$, $n$ and $B$ for CSIDH-3 and CSIDH-5 were not given in the paper, and are produced here following their methodology. Message size is the number of bytes to represent a $j$-invariant, and private key size is the space required to store the exponent vector $\mathbf{e} \in \mathbb{Z}^n$.

For our signature schemes we may work with less specific primes than considered in CSIDH [10]. For example, CSIDH takes $p+1 = 4\prod_{i=1}^{n} \ell_i$, whereas we may be able to use fewer primes and just multiply by a random co-factor to get a large enough $p$.

### 2.3 Public key signature schemes

One can describe Fiat-Shamir-type signatures in various ways, including the language of sigma protocols or identification schemes. In the main body of our paper we mostly work with the language of signatures, and give proofs directly in this formulation. In Section B.1 we use the language of identification schemes, and introduce the terminology fully there.

A *canonical identification scheme* consists of algorithms $(\mathsf{KeyGen}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ and a set $\mathsf{ChSet}$. The randomised algorithm $\mathsf{KeyGen}(1^\lambda)$ outputs a key pair $(pk, sk)$. The deterministic algorithm $\mathsf{P}_1$ takes $sk$ and randomness $r_1$ and computes $(W, \mathsf{St}) = P_1(sk, r_1)$. Here $\mathsf{St}$ denotes state information to be passed to $P_2$. A challenge $c$ is sampled uniformly from $\mathsf{ChSet}$. The deterministic algorithm $P_2$ then computes $Z = P_2(sk, W, c, \mathsf{St}, r_2)$ or $\perp$, where $r_2$ is the randomness. The output $\perp$ corresponds to an abort in the "Fiat-Shamir with aborts" paradigm. We require that $\mathsf{V}(pk, W, c, Z) = 1$ for a correctly formed transcript $(W, c, Z)$.

A *public key signature scheme* consists of algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}$. The randomised algorithm $\mathsf{KeyGen}(1^\lambda)$ outputs a pair $(pk, sk)$, where $\lambda$ is a security parameter. The randomised algorithm $\mathsf{Sign}$ takes input the private key $sk$ and a message $\mathsf{msg}$, and outputs $\sigma = \mathsf{Sign}(sk, \mathsf{msg})$. The verification algorithm $\mathsf{Verify}(pk, \mathsf{msg}, \sigma)$ returns 0 or 1. We require $\mathsf{Verify}(pk, \mathsf{msg}, \mathsf{Sign}(sk, \mathsf{msg})) = 1$.

The Fiat-Shamir transform is a construction to turn a canonical identification scheme into a public key signature scheme. The main idea is to make the interactive identification scheme into a non-interactive scheme by replacing the challenge $c$ by a hash $H(W, \mathsf{msg})$.

The standard notion of security is *unforgeability against chosen-message attack (UF-CMA)*. A UF-CMA adversary against the signature scheme is a randomised polynomial-time algorithm $A$ that plays the following game against a challenger. The challenger runs $\mathsf{KeyGen}$ to get $(pk, sk)$ and runs $A(pk)$. The adversary $A$ sends messages $\mathsf{msg}$ to the challenger, and receives $\sigma = \mathsf{Sign}(sk, \mathsf{msg})$ in return. The adversary outputs $(\mathsf{msg}^*, \sigma^*)$ and wins if $\mathsf{Verify}(pk, \mathsf{msg}^*, \sigma^*) = 1$ and if $\mathsf{msg}^*$ was not one of the messages previously sent by the adversary to the challenger. A signature scheme is UF-CMA secure if there is no polynomial-time adversary that wins with non-negligible probability.

## 3 Basic Signature Scheme

This section contains our main ideas and presents a basic signature scheme. We focus in this section on classical adversaries and proofs in the random oracle model. Hence our signature is based on the traditional Fiat-Shamir transform. For schemes and analysis against a post-quantum adversary see Appendix B.

## 3.1 Stolbunov's scheme

Section 2.B of Stolbunov's PhD thesis [38] contains a sketch of a signature scheme based on isogeny problems (though his description is not complete and he does not give a proof of security). It is a Fiat-Shamir scheme based on an identification protocol. Section 4 of Couveignes [14] also sketches the identification protocol, but does not mention signature schemes.

The public key consists of $E$ and $E_A = \mathfrak{a} * E$, where $\mathfrak{a} = \prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ is the private key. To construct the private key one uniformly chooses an exponent vector $\mathbf{e} = (e_1, \ldots, e_n) \in [-B, B]^n \subseteq \mathbb{Z}^n$ for some suitably chosen constant $B$. Stolbunov assumes the relation lattice for the ideal class group is known, and uses it in Section 2.6.1 to sample ideal classes uniformly at random. Section 2.6.2 of [38] suggests an approach to approximate the uniform distribution.

In the identification protocol the prover generates $t$ random ideals $\mathfrak{b}_k = \prod_{i=1}^{n} \mathfrak{l}_i^{f_{k,i}}$ for $1 \leq k \leq t$ and computes $\mathcal{E}_k = \mathfrak{b}_k * E$. Here the exponent vectors $\mathbf{f}_k = (f_{k,1}, \ldots, f_{k,n})$ are uniformly and independently sampled in a region like $[-B, B]^n$ (Stolbunov assumes these ideal classes are uniformly sampled). The prover sends $(j(\mathcal{E}_k) : 1 \leq k \leq t)$ to the verifier. The verifier responds with $t$ uniformly chosen challenge bits $b_1, \ldots, b_t \in \{0, 1\}$. If $b_k = 0$ the prover responds with $\mathbf{f}_k = (f_{k,1}, \ldots, f_{k,n})$ and the verifier checks that $j(\mathcal{E}_k) = j((\prod_{i=1}^{n} \mathfrak{l}_i^{f_{k,i}}) * E)$. If $b_k = 1$ the prover responds with a representation of $\mathfrak{b}_k \mathfrak{a}^{-1}$. When $b_k = 1$ the verifier checks that $j(\mathcal{E}_k) = j((\mathfrak{b}_k \mathfrak{a}^{-1}) * E_A)$. A cheating prover (who does not know the private key) can succeed with probability $1/2^t$.

The major problem with the above idea is how to represent the ideal class of $\mathfrak{b}_k \mathfrak{a}^{-1}$ in a way that does not leak $\mathfrak{a}$. Stolbunov notes that sending the vector $\mathbf{f}_k - \mathbf{e} = (f_{k,i} - e_i)$ would not be secure as it would leak the private key, instead he mentions in one sentence a solution that is applicable when the discrete logs of the primes $\mathfrak{l}_i$ in the class group are known. Couveignes also does not explain how to prevent this leakage. A main contribution of our paper is to give solutions to this problem.

To obtain a signature scheme Stolbunov applies the Fiat-Shamir transform, and hence obtains the challenge bits $b_k$ as the hash value $H(j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t), \mathsf{msg})$ where $H$ is a cryptographic hash function with $t$-bit output and $\mathsf{msg}$ is the message to be signed. The signature consists of the binary string $b_1 \cdots b_t$ and the representations of the ideal classes $\mathfrak{b}_k$ when $b_k = 0$ and $\mathfrak{b}_k \mathfrak{a}^{-1}$ when $b_k = 1$.

The verifier computes, for $1 \leq k \leq t$, $\mathcal{E}_k = \mathfrak{b}_k * E$ when $b_k = 0$ and $\mathcal{E}_k = \mathfrak{b}_k \mathfrak{a}^{-1} * E_A$ when $b_k = 1$. The verifier then computes $H(j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t), \mathsf{msg})$ and checks whether this is equal to the binary string $b_1 \cdots b_t$, and accepts the signature if and only if the strings agree.

We stress that neither Couveignes nor Stolbunov give a secure post-quantum signature scheme. Both authors assume that the relations in the ideal class group have been computed (Stolbunov needs this to prevent leakage, since he needs to know discrete logs). However the cost to compute the relations in the ideal class group on a classical computer is roughly the same as the cost to break the scheme on a quantum computer (using the Kuperberg or Regev algorithms). Hence it does not make sense to assume the generator of the scheme knows the relations in the ideal class group (although see Appendix C for a version of this idea in the fully post-quantum setting where quantum computers are readily available).

The main contribution of this paper is to give a solution to the problem of representing $\mathfrak{b}_k \mathfrak{a}^{-1}$ without leaking the private key and without needing to compute relations in the ideal class group. Our solution uses ideas from lattice cryptography (Fiat-Shamir with aborts) and we describe this in the remainder of the section. An alternative solution requires a basis of a relation lattice in the ideal class group, which can be computed efficiently using a quantum computer. We describe this alternative solution in Appendix C.

## 3.2 Using rejection sampling

The idea is to use rejection sampling in exactly the way proposed by Lyubashevsky [29] in the context of lattice signatures.

Let $B > 0$ be a constant. When generating the private key we sample uniformly $e_i \in [-B, B]$ for $1 \leq i \leq n$. Let $\mathbf{e} = (e_1, \ldots, e_n)$. The value $B$ may be chosen large enough that $\prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ covers most ideal classes and so that the output distribution is close to uniformly distributed in $\mathrm{Cl}(\mathcal{O})$, but we avoid any explicit requirement or assumption that this distribution is uniform. We refer to Definition 1 for more discussion of this issue, and in Appendix B we consider a variant where the ideals are definitely not distributed uniformly in $\mathrm{Cl}(\mathcal{O})$.

Exponents $f_{k,i}$ are sampled uniformly in $[-(nt+1)B, (nt+1)B]$, where $t$ is the number of parallel rounds of the identification/signature protocol and $n$ is the number of primes. Let $\mathbf{f}_k = (f_{k,1}, \ldots, f_{k,n})$, $\mathfrak{b}_k = \prod_{i=1}^{n} \mathfrak{l}_i^{f_{k,i}}$ and define $\mathcal{E}_k = \mathfrak{b}_k * E$.

If the $k$-th challenge bit $b_k$ is zero then the prover responds with $\mathbf{f}_k = (f_{k,1}, \ldots, f_{k,n})$ and the verifier checks that $j(\mathcal{E}_k) = j((\prod_{i=1}^{n} \mathfrak{l}_i^{f_{k,i}}) * E)$ as in the basic scheme above.[3] If $b_k = 1$ then the prover is required to provide a representation of $\mathfrak{b}_k\mathfrak{a}^{-1}$, the idea is to compute the vector $\mathbf{z}_k = (z_{k,1}, \ldots, z_{k,n})$ defined by $z_{k,i} = f_{k,i} - e_i$ for $1 \le i \le n$. As already noted, outputting $\mathbf{z}$ directly would potentially leak the secret. To prevent this leakage we only output $\mathbf{z}_k$ if all its entries satisfy $|z_{k,i}| \le ntB$. We give the signature scheme in Figure 1. It remains to show that in the accepting case the vector leaks no information about the private key, and that the rejecting case occurs with low probability. We do this in the following two lemmas.

**Lemma 1.** *The distribution of vectors $\mathbf{z}_k$ output by the signing algorithm is the uniform distribution and therefore is independent of the private key $\mathbf{e}$.*

*Proof.* Let $U = [-(nt+1)B, (nt+1)B]$. Then $\#U = 2(nt+1)B + 1$. If $e \in [-B, B]$ then

$$[-ntB, ntB] \subseteq U - e = \{f - e : f \in U\} \subseteq [-(nt+2)B, (nt+2)B].$$

Hence, when rejection sampling (only outputting values $f_{k,i} - e_i$ in the range $[-ntB, ntB]$) is applied then the output distribution of $\mathbf{z}_k$ is the uniform distribution on $[-ntB, ntB]^n$. This argument does not depend on the choice of $\mathbf{e}$, so the output distribution is independent of $\mathbf{e}$. □

**Lemma 2.** *The probability that the signing algorithm outputs a signature (i.e., does not output $\perp$) is at least $1/e > 1/3$.*

*Proof.* Let notation be as in the proof of Lemma 1. For fixed $e \in [-B, B]$ and uniformly sampled $f \in U = [-(nt+1)B, (nt+1)B]$, the probability that a value $f - e$ lies in $[-ntB, ntB]$ is

$$\frac{2ntB+1}{2(nt+1)B+1} = 1 - \frac{2B}{2(nt+1)B+1} \ge 1 - \frac{1}{nt+1}.$$

Hence, the probability that all of the values $z_{k,i}$ over $1 \le k \le t, 1 \le i \le n$ lie in $[-ntB, ntB]$ is at least $(1 - 1/(nt+1))^{nt}$. Using the inequality $1 - 1/(x+1) \ge e^{-1/x}$ for $x \ge 1$ it follows that the probability that all values are in the desired range is at least

$$\left(e^{-1/nt}\right)^{nt} = e^{-1}.$$

This completes the proof. □

We can therefore get a rough idea of parameters and efficiency for the scheme. Let $\lambda$ be a security parameter (e.g., $\lambda = 128$ or $\lambda = 256$), for security we need at least $t = \lambda$ so that an attacker cannot guess the hash value or invert the hash function (see also the proof of Theorem 1). We also need a large enough set of private keys, so we need $(2B+1)^n$ large enough. The signature contains one hash value of $t$ bits, plus $t$ vectors $\mathbf{f}_k$ or $\mathbf{z}_k$ with entries of size bounded by $(nt+1)B$, for a total of $\lambda + t\lceil n \log(2(nt+1)B+1) \rceil$ bits (assuming each vector is represented optimally). If we take $t = \lambda = 128$, and $(n, B) = (74, 5)$ as in CSIDH-1, we obtain signatures of 19.6 kilobytes (see also Table 2).

To sign/verify one needs to evaluate the action of either of $\mathfrak{b}_k$ and $\mathfrak{b}_k\mathfrak{a}^{-1}$ for every $1 \le k \le t$, which means that for each $k$ and each prime $\mathfrak{l}_i$ one needs to compute up to $ntB$ isogenies of degree $\ell_i$. Hence, the total number of isogeny computations is upper bounded by $(nt)^2B$. The quadratic dependence on $nt$ is a major inconvenience.

[3] In the scheme and analysis I actually apply rejection sampling to the case $b_k = 0$. It doesn't really matter one way or the other.

**Algorithm 1** KeyGen

```
Input: $B$, $\mathfrak{l}_1, \ldots \mathfrak{l}_n$, $E$
Output: $sk = \mathbf{e}$ and $pk = E_A$
1: $\mathbf{e} \leftarrow [-B, B]^n$
2: $E_A = (\prod_{i=1}^{n} \mathfrak{l}_i^{e_i}) * E$
3: return $sk = \mathbf{e}$, $pk = E_A$
```

**Algorithm 2** Sign

```
Input: $\mathsf{msg}$, $(E, E_A)$, $\mathbf{e}$
Output: $(\mathbf{z}_1, \ldots, \mathbf{z}_t)$, $(b_1, \ldots, b_t)$
1: for $k = 1, \ldots, t$ do
2:     $\mathbf{f}_k \leftarrow [-(nt+1)B, (nt+1)B]^n$
3:     $\mathcal{E}_k = (\prod_{i=1}^{n} \mathfrak{l}_i^{f_{k,i}}) * E$
4: end for
5: $b_1 \| \cdots \| b_t = H(j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t), \mathsf{msg})$
6: for $k = 1, \ldots, t$ do
7:     if $b_k = 0$ then
8:         $\mathbf{z}_k = \mathbf{f}_k$
9:     else
10:        $\mathbf{z}_k = \mathbf{f}_k - \mathbf{e}$
11:    end if
12:    if $\mathbf{z}_k \notin [-ntB, ntB]^n$ then
13:        return $\perp$
14:    end if
15: end for
16: return $\sigma = (\mathbf{z}_1, \ldots, \mathbf{z}_t, b_1, \ldots, b_t)$
```

**Algorithm 3** Verify

```
Input: $\mathsf{msg}$, $(E, E_A)$, $\sigma$
Output: Valid/Invalid
1: Parse $\sigma$ as $(\mathbf{z}_1, \ldots, \mathbf{z}_t, b_1, \ldots, b_t)$
2: for $k = 1, \ldots, t$ do
3:     if $b_k = 0$ then
4:         $\mathcal{E}_k = (\prod_{i=1}^{n} \mathfrak{l}_i^{z_{k,i}}) * E$
5:     else
6:         $\mathcal{E}_k = (\prod_{i=1}^{n} \mathfrak{l}_i^{z_{k,i}}) * E_A$
7:     end if
8: end for
9: $b'_1 \| \cdots \| b'_t = H(j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t), \mathsf{msg})$
10: if $(b'_1, \ldots, b'_t) = (b_1, \ldots, b_t)$ then
11:     return Valid
12: else
13:     return Invalid
14: end if
```

**Fig. 1.** The basic signature scheme using rejection sampling.

For example, taking $(n, t, B) = (74, 128, 5)$ gives around $2^{28}$ isogeny computations in signature/verification. We can make $t$ small using the techniques in later sections, but one needs $n$ large unless $B$ is going to get very large. So even going down to $t = 8$ still has signatures requiring around $2^{20}$ isogeny computations. The acceptance probability estimate from Lemma 2 is very close to the true value: for $(n, t, B) = (74, 128, 5)$ then the true acceptance probability is approximately $0.36790$, while $e^{-1} \approx 0.36788$.

We discuss some possible optimisations in Appendix A, including the idea to use discrete Gaussians instead of uniform distributions for the vectors.

## 3.3 Security proof

We now prove security of the basic scheme in the random oracle model against a classical adversary. The proof technique is the standard approach that uses the forking lemma. In this section we do not consider quantum adversaries, or give a proof in the quantum random oracle model (QROM). A proof in the QROM follows from the approach in Appendix B.

First we need to discuss some subtleties about the distribution of ideal classes coming from the key generation and signing algorithms.

**Definition 1.** *Fix distinct ideals $\mathfrak{l}_1, \dots, \mathfrak{l}_n$. For $B \in \mathbb{N}$, consider the random variable $\mathfrak{a}$ which is the ideal class of $\prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ over a uniformly random $\mathbf{e} \in [-B, B]^n$. Define $\mathcal{D}_B$ to be the distribution on $\mathrm{Cl}(\mathcal{O})$ corresponding to this random variable. Define $M_B$ to be an upper bound on the probability, over $\mathfrak{a}, \mathfrak{b}$ sampled from $\mathcal{D}_B$, that $\mathfrak{a} \equiv \mathfrak{b}$.*

In other words, $\mathcal{D}_B$ is the output distribution of the public key generation algorithm. Understanding the distribution $\mathcal{D}_B$ is non-trivial in general.[4] For small $B$ and $n$ (so that $(2B+1)^n \ll \#\mathrm{Cl}(\mathcal{O})$) we expect $\mathcal{D}_B$ to be the uniform distribution on a subset of $\mathrm{Cl}(\mathcal{O})$ of size $(2B+1)^n$. For fixed $n$ and large enough $B$ it should be the case that $\mathcal{D}_B$ is very close to the uniform distribution on $\mathrm{Cl}(\mathcal{O})$. A full study of the distribution $\mathcal{D}_B$ is beyond the scope of this paper, but is a good problem for future work.

For the isogeny problem to be hard for public keys we certainly need $M_B \leq 1/2^{\lambda}$, where $\lambda$ is the security parameter. In the proof we will need to use $M_{ntB}$, since the concern is about the auxiliary curves generated during the signing algorithm. We do not require these curves to be uniformly sampled, but in practice we can certainly assume that $M_{ntB} = O(1/\sqrt{p})$. In any case, it is negligible in the security parameter.

*Problem 2.* Let notation be as in the key generation protocol of the scheme. Given $(E, E_A)$, where $E_A = \mathfrak{a} * E$ for some ideal $\mathfrak{a} = \prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ and where the exponent vector $\mathbf{e} = (e_1, \dots, e_n)$ is uniformly sampled in $[-B, B]^n \subseteq \mathbb{Z}^n$, to compute any ideal equivalent to $\mathfrak{a}$.

Depending on how close to uniform is the distribution $\mathcal{D}_B$, this problem may or may not be equivalent to Problem 1 and may or may not have a random self-reduction. Nevertheless, we believe this is a plausible assumption.

We recall the forking lemma, in the formulation of Bellare and Neven [4].

**Lemma 3.** *(Bellare and Neven [4]) Fix an integer $Q \geq 1$. Let $A$ be a randomized algorithm that takes as input $h_1, \dots, h_Q \in \{0,1\}^t$ and outputs $(J, \sigma)$ where $1 \leq J \leq Q$ with probability $\wp$. Consider the following experiment: $h_1, \dots, h_Q$ are chosen uniformly at random in $\{0,1\}^t$; $A(h_1, \dots, h_Q)$ returns $(I, \sigma)$ such that $I \geq 1$; $h'_I, \dots, h'_Q$ are chosen uniformly at random in $\{0,1\}^t$; $A(h_1, \dots, h_{I-1}, h'_I, \dots, h'_Q)$ returns $(I', \sigma')$. Then the probability that $I' = I$ and $h'_I \neq h_I$ is at least $\wp(\wp/Q - 1/2^t)$.*

**Theorem 1.** *In the random oracle model, the basic signature scheme of Figure 1 is unforgeable under a chosen message attack under the assumption that Problem 2 is hard.*

[4] Even the analogous problem of understanding the distribution of $\prod_i \ell_i^{e_i} \pmod{q}$, where $\ell_i$ are small primes and $q$ is some integer, is an open problem in general.

*Proof.* Consider a polynomial-time adversary $A$ against the signature scheme. So $A$ takes a public key, makes queries to the hash function $H$ and the signing oracle, and outputs a forgery of a signature with respect the public key.

Let $(E, E_A = \mathfrak{a} * E)$ be an instance of Problem 2. The simulator runs the adversary $A$ with public key $(E, E_A)$.

Suppose the adversary $A$ makes at most $Q$ (polynomial in the security parameter) queries in total to either the random oracle $H$ or the signing oracle. We now explain how the simulator responds to these queries. The simulator maintains a list, initially empty, of pairs $(x, H(x))$ for each value of the random oracle that has been defined.

**Sign queries:** To answer a Sign query on message $\mathsf{msg}$ the simulator chooses $t$ uniformly chosen bits $b_1, \dots, b_t \in \{0,1\}$. When $b_k = 0$ the simulator randomly samples $z_k \leftarrow [-ntB, ntB]^n$ and sets $\mathfrak{b}_k = \prod_{i=1}^n \mathfrak{l}_i^{z_{k,i}}$ and computes $\mathcal{E}_k = \mathfrak{b}_k * E$, just like in the real signing algorithm. When $b_k = 1$ the simulator chooses a random ideal $\mathfrak{c}_k = \prod_{i=1}^n \mathfrak{l}_i^{z_{k,i}}$ for $z_{k,i} \in [-ntB, ntB]$ and computes $\mathcal{E}_k = \mathfrak{c}_k * E_A$. By Lemma 1, the values $j(\mathcal{E}_k)$ and $\mathbf{z}_k$ are distributed exactly as in the real signing algorithm. We program the random oracle (update the hash list) so that $H(j(\mathcal{E}_1), \dots, j(\mathcal{E}_t), \mathsf{msg}) := b_1 \cdots b_t$, unless the random oracle has already been defined on this input in which case the simulation fails and outputs $\perp$. The probability of failure is at most $Q/M_{ntB}^t$, where $M_{ntB}$ is defined in Definition 1 to be an upper bound on the probability of a collision in the sampling of ideal classes. Note that $Q/M_{ntB}^t$ is negligible. Assuming the simulation does not fail, the output is a valid signature and is indistinguishable from signatures output by the real scheme in the random oracle model.

**Hash queries:** To answer a random oracle query on input $x$ the simulator checks if $(x, y)$ already appears in the list, and if so returns $y$. Otherwise the simulator chooses uniformly at random $y \in \{0,1\}^t$ and sets $H(x) := y$ and adds $(x, y)$ to the list.

Eventually $A$ outputs a forgery $(\mathsf{msg}, \sigma = (\mathbf{z}_1, \dots, \mathbf{z}_t, b_1 \cdots b_t))$ that passes the verification equation. Define $\mathfrak{c}_k = \prod_i \mathfrak{l}_i^{z_{k,i}}$. The proof now invokes the Forking Lemma (see Bellare-Neven [4]). The adversary is replayed with the same random tape and the exact same simulation, except that one of the hash queries is answered with a different binary string. With non-negligible probability the adversary outputs a forgery $\sigma = (\mathbf{z}'_1, \dots, \mathbf{z}'_t, b'_1 \cdots b'_t)$ for the same message $\mathsf{msg}$ and the same input $(j(\mathcal{E}_1), \dots, j(\mathcal{E}_t), \mathsf{msg})$ to $H$, but a different output string $b'_1 \cdots b'_t$. Let $k$ be an index such that $b_k \neq b'_k$ (without loss of generality $b_k = 0$ and $b'_k = 1$). Then the ideal classes $\mathfrak{c}_k$ and $\mathfrak{c}'_k$ in the two signatures are such that $j(\mathfrak{c}_k * E) = j(\mathfrak{c}'_k * E_A)$ and so $\mathfrak{c}'_k \mathfrak{c}_k^{-1} = \prod_i \mathfrak{l}_i^{z'_{k,i} - z_{k,i}}$ is a solution to the problem instance. □

We make two observations about the use of the forking lemma. First, as always, the proof is not tight since if the adversary succeeds with probability $\epsilon$ then the simulator solves the computational problem with probability proportional to $\epsilon^2$. Second, the hash output length $t$ in Lemma 3 only appears in the term $1/2^t$, so it suffices to take $t = \lambda$. There may be situations where a larger hash output is needed; for more discussion about hash output sizes we refer to Neven, Smart and Warinschi [33].

## 4 Smaller sigs

The signature size of the basic scheme is very large, since the sigma protocol that underlies the identification scheme only has single bit challenges. In practice we need $t \geq 128$, which means signatures are very large. To get shorter signatures it is natural to try to increase the size of the challenges. In this section we sketch an approach to obtain $s$-bit challenge values for any small integer $s \in \mathbb{N}$, by trading the challenge size with the public key size. In the next section we explain how to shorten the public keys again.

The basic idea is to have public keys $(E_{A,0} = \mathfrak{a}_0 * E, \dots, E_{A,2^s-1} = \mathfrak{a}_{2^s-1} * E)$. For each $0 \leq m < 2^s$ we choose $\mathbf{e}_m \leftarrow [-B, B]^n$ and set $E_{A,m} = (\prod_{i=1}^n \mathfrak{l}_i^{e_{m,i}}) * E$. The signing algorithm for user A chooses $t$ random ideals $\mathfrak{b}_k = \prod_{i=1}^n \mathfrak{l}_i^{f_{k,i}}$ and computes $\mathcal{E}_k = \mathfrak{b}_k * E$, as before. Now we have $s$-bit challenges $b_1, \dots, b_t \in \{0, 1, \dots, 2^s - 1\}$. For each $1 \leq k \leq t$ the signer computes $\mathbf{z}_k = \mathbf{f}_k - \mathbf{e}_{b_k}$, which corresponds to the ideal class $\mathfrak{c}_k = \mathfrak{b}_k \mathfrak{a}_{b_k}^{-1}$ and the verifier can check that $j(\mathcal{E}_k) = j(\mathfrak{c}_k * E_{A,b_k})$.

A signature consists of one hash value, plus $t$ vectors $\mathbf{z}_k$ with entries of size bounded by $ntB$, i.e., a total of $\lambda + t\lceil n \log(2ntB + 1)\rceil$ bits, similar to the previous section. But now for security we now only require $ts \geq \lambda$. Taking, say, $\lambda = 128$ and $s = 16$ can mean $t$ as low as 8, and so only 8 vectors need to be transmitted as part of the signature, giving signatures of well under one kilobyte (see Table 3). Of course the public key now includes $2^{16}$ $j$-invariants (elements of $\mathbb{F}_p$) which would be around 4 megabytes, and key generation is also $2^{16}$ times slower.

As far as we can tell, this idea cannot be applied to the schemes of Yoo et al [42] or Galbraith et al [22].

### 4.1 Security

A trivial modification to the proof of Theorem 1 can be applied in this setting. But note that the forking lemma produces two signatures such that $b_k \neq b'_k$ for some index $k$. Hence from a successful forger we derive two ideal classes $\mathfrak{c}_k$ and $\mathfrak{c}'_k$ such that $j(\mathfrak{c}_k * E_{A,b_k}) = j(\mathfrak{c}'_k * E_{A,b'_k})$. It follows that $(\mathfrak{c}'_k)^{-1}\mathfrak{c}_k$ is an ideal class corresponding to an isogeny $E_{A,b_k} \to E_{A,b'_k}$. Hence the computational assumption underlying the scheme is the following.

*Problem 3.* Let notation be as in the key generation protocol of the scheme. Consider a set of $2^s$ elliptic curves $\{E_{A,0}, \dots, E_{A,2^s-1}\}$, all of the form $E_{A,m} = \mathfrak{a}_m * E$ for some ideal $\mathfrak{a}_m = \prod_{i=1}^{n} \mathfrak{l}_i^{e_{m,i}}$ where the exponent vectors $\mathbf{e}_m$ are uniformly sampled in $[-B, B]^n \subseteq \mathbb{Z}^n$. The "one out of $2^s$" isogeny problem is to compute an ideal corresponding to any isogeny $E_{A,m} \to E_{A,m'}$ for some $m \neq m'$.

We believe this problem is hard for classical and quantum computers. One can easily obtain a non-tight reduction of this problem to Problem 2. However, if the ideals $\mathfrak{a}_m$ are not sampled uniformly at random from $\mathrm{Cl}(\mathcal{O})$ then we do not know how to obtain a random-self-reduction for this problem, which prevents us from having a tight reduction to Problem 2.

**Theorem 2.** *In the random oracle model, the signature scheme of this section is unforgeable under a chosen message attack under the assumption that Problem 3 is hard.*

The proof of this theorem is almost identical to the proof of Theorem 1 and so is omitted.

### 4.2 Variant based on a more natural problem

Problem 3 is a little un-natural. It would be more pleasing to prove security based on Problem 1 or Problem 2. We now explain that one can prove security based on Problem 1, under an assumption about uniform sampling of ideal classes.

Suppose in this section that the distribution $\mathcal{D}_B$ of Definition 1 has negligible statistical distance (Renyi divergence can also be used here) from the uniform distribution. This assumption is reasonable for bounded $n$ and very large $B$; but we leave for future work to determine whether practical parameters for isogeny crypto can be obtained under this constraint.

**Lemma 4.** *Let parameters be such that the statistical distance between $\mathcal{D}_B$ and the uniform distribution on $\mathrm{Cl}(\mathcal{O})$ is negligible. Suppose that all the prime ideals $\mathfrak{l}_i$ have norm bounded as $O(\log(p))$ Then given an algorithm that runs in time $T$ and solves Problem 3 with probability $\epsilon$, there is an algorithm to solve Problem 1 with time $T + O(2^s \log(p)^5)$ and success probability $\epsilon/2$.*

*Proof.* Let $A$ be an algorithm for Problem 3, and let $(E, E_A = \mathfrak{a} * E)$ be an instance of Problem 1.

Choose random ideal classes $\mathfrak{b}_0, \dots, \mathfrak{b}_{2^s-1}$ (chosen as $\mathfrak{b}_m = \prod_{i=1}^{n} \mathfrak{l}_i^{u_{i,m}}$ for $0 \leq m < 2^s$ and $u_{i,m} \in [-B, B]$) and compute $E'_{A,m} = \mathfrak{b}_m * E$ for $0 \leq m < 2^{s-1}$ and $E'_{A,m} = \mathfrak{b}_m * E_A$ for $2^{s-1} \leq m < 2^s$. Choose a random permutation $\pi$ on $\{0, 1, \dots, 2^s - 1\}$ and construct the sequence $E_{A,m} = E'_{A,\pi(m)}$. This computation takes $O(2^s \log(p)^5)$ bit operations, since $n$ and $B$ and the norm $\ell_i$ of $\mathfrak{l}_i$ are all $O(\log(p))$. Note that these curves are all uniformly sampled in the isogeny class, and so there is no way to distinguish whether any individual curve has been generated from $E$ or $E_A$. This is where the subtlety about distributions appears: it is crucial that the curves derived from the pair $(E, E_A)$ are indistinguishable from the curves in Problem 3.

Now run the algorithm $A$ on this input. Since the input is indistinguishable from a real input, $A$ runs in time $T$ and succeeds with probability $\epsilon$. In the case of success, we have an ideal $\mathfrak{c}$ corresponding to an isogeny $E_{A,m} \to E_{A,m'}$ for some $m \neq m'$. With probability $1/2$ we have that one of the curves, say $E_{A,m}$, is known to the simulator as $\mathfrak{b} * E$ and the other (i.e., $E_{A,m'}$) is known as $\mathfrak{b}' * E_A$. If this event occurs then we have $\mathfrak{c}\mathfrak{b} * E = \mathfrak{b}' * E_A$ (or vice versa) in which case $\mathfrak{c}\mathfrak{b}(\mathfrak{b}')^{-1}$ is a solution to the original instance. $\square$

Note that this proof introduces an extra $1/2$ factor in the success probability, but this is not a serious issue since the security proof isn't tight anyway.

Using this result, the following theorem is an immediate consequence of Theorem 2.

**Theorem 3.** *Let parameters be such that the statistical distance between $\mathcal{D}_B$ and the uniform distribution on $\mathrm{Cl}(\mathcal{O})$ is negligible. In the random oracle model, the signature scheme of this section is unforgeable under a chosen message attack under the assumption that Problem 1 is hard.*

We have a tight proof in Appendix B based on a less standard assumption. It is an open problem to have a tight proof and also the security based on Problem 1.

### 4.3 Reducing storage for private keys

Rather than storing all the private keys $\mathfrak{a}_m$ for $0 \le m < 2^s$ one could have generated them using a pseudorandom function as $\mathsf{PRF}(\mathsf{seed}, i)$ where seed is a seed and $i$ is used to generate the $i$-th private key (which is an integer exponent vector). The prover only needs to store seed and can then recompute the private keys as needed. Of course, during key generation one needs to compute all the public keys, but during signing one only needs to determine $t \approx 8$ private keys (although this adds a cost to the signing algorithm).

## 5 Smaller public keys

The approach of Section 4 gives signatures that are potentially quite small, but at the expense of very large public keys. In some settings (e.g., software signing or licence checks) large public keys can be easily accommodated, while in other settings (e.g., certificate chains) it makes no sense to shorten signatures at the expense of public key size. In this section we explain how to use techniques from hash-based signatures to compress the public key while also maintaining compact signatures. The key idea is to use a Merkle tree [31] with leaves the public curves $E_{A,0}, \ldots, E_{A,2^s-1}$, and use the tree root (a single hash value) as public key. However, the security of plain Merkle trees depends on collision resistance of the underlying hash function, thus requiring hashes of size at least twice the security parameter. Instead, we use a modified Merkle tree, as introduced in the hash-based signatures XMSS-T [24] and SPHINCS+ [5], whose security relies on the second preimage resistance of a keyed hash function.

Let $\lambda$ be a security parameter, and let $n, B, s, t, p$ be as in the previous sections; we assume that $\lceil \log p \rceil > 2\lambda$, as this is the case in any secure instantiation. Let the following (public) functions be given:

- $\mathsf{PRF}_{\text{secret}} : \{0,1\}^\lambda \times \{0,1\}^s \to [-B, B]^n$,
- $\mathsf{PRF}_{\text{key}} : \{0,1\}^\lambda \times \{0,1\}^{s+1} \to \{0,1\}^\lambda$,
- $\mathsf{PRF}_{\text{mask}} : \{0,1\}^\lambda \times \{0,1\}^{s+1} \to \{0,1\}^{\lceil \log p \rceil}$ three pseudo-random functions, and
- $M : \{0,1\}^\lambda \times \{0,1\}^{\lceil \log p \rceil} \to \{0,1\}^\lambda$ a keyed hash function (where we think of the first $\lambda$ bits as the key and the second $\lceil \log p \rceil$ bits as the input).

Finally, let $\mathsf{PK.seed}$ and $\mathsf{SK.seed}$ be two random seeds; as the names suggest, $\mathsf{PK.seed}$ is part of the public key, while $\mathsf{SK.seed}$ is part of the secret key. Like in Section 4.3, we define the secret ideals $\mathfrak{a}_m = \prod_{i=1}^n \mathfrak{l}_i^{e_{m,i}}$, where $\mathbf{e}_m = \mathsf{PRF}_{\text{secret}}(\mathsf{SK.seed}, m)$, and the public curves $E_{A,m} = \mathfrak{a}_m * E$, for $0 \le m < 2^s$.

We set up a hash tree by defining $h_{l,u}$ for $0 \le l \le s$ and $0 \le u < 2^{s-l}$. First we set

$$h_{s,u} = M\big(\mathsf{PRF}_{\text{key}}(\mathsf{PK.seed}, 2^s + u),\ j(E_{A,u}) \oplus \mathsf{PRF}_{\text{mask}}(\mathsf{PK.seed}, 2^s + u)\big)$$

for $0 \le u < 2^s$, where $\oplus$ denotes bitwise XOR. Now, for any $0 \le l < s$, the rows of the hash tree are defined as

$$h_{l,u} = M\big(\mathsf{PRF}_{\text{key}}(\mathsf{PK.seed}, 2^l + u),\ (h_{l+1,2u} \| h_{l+1,2u+1}) \oplus \mathsf{PRF}_{\text{mask}}(\mathsf{PK.seed}, 2^l + u)\big).$$

Finally, the public key is set to the pair $(\mathsf{PK.seed}, h_{0,0})$.

To prove that a value $E_{A,u}$ is in the hash tree, we use its *authentication path*. That is the list of the hash values $h_{l,u'}$, for $1 \leq l \leq s$, occurring as siblings of the nodes on the path from $h_{s,u}$ to the root. The proof in [24, Appendix B] shows that having $M$ output $\lambda$-bit hashes gives a (classical) security of approximately $2^{\lambda}$. See [24,5] for more details.

Typically, in hash-based signatures the secret key would only contain SK.seed, since all secret and public values can be reconstructed from it at an acceptable cost. However, in our case recomputing the leaves of the hash tree ($2^s$ class group actions) is much more expensive than recomputing the internal nodes ($2^s - 1$ hash function evaluations), thus we set the secret key to the tuple $(\mathsf{SK.seed}, h_{s,0}, \dots, h_{s,2^s-1})$. This is a considerably large secret key, e.g., around 1 megabyte when $\lambda = 128$ and $s = 16$, but it is offset by a more than tenfold gain in signing time. Also note that the values $h_{s,u}$ can (and will) be leaked without any loss in security, they are indeed part of the uncompressed public key, thus they are more formally treated as auxiliary signer data, rather than as part of the secret key.

To sign we proceed like in Section 4, but the signature now needs to contain additional information. The signer computes the random ideals $\mathfrak{b}_1, \dots, \mathfrak{b}_t$ and the associated curves $\mathcal{E}_1, \dots, \mathcal{E}_t$ to obtain the challenges $b_1, \dots, b_t$. Then, using $\mathsf{PRF}_{\text{secret}}$, they obtain the secrets $\mathfrak{a}_{b_1}, \dots, \mathfrak{a}_{b_t}$, recompute the public curves $E_{A,b_1}, \dots, E_{A,b_t}$, and the ideals $\mathfrak{c}_i = \mathfrak{a}_{b_i}^{-1}\mathfrak{b}_i$. The signature is made of the ideals $\mathfrak{c}_1, \dots, \mathfrak{c}_t$, the curves $E_{A,b_1}, \dots, E_{A,b_t}$, and their authentication paths in the hash tree. The verifier computes $\mathcal{E}_i$ as $\mathfrak{c}_i * E_{A,b_i}$, obtains the challenges $b_1, \dots, b_t$, and uses them to verify the authentication paths. Hence, the signature contains $t$ ideals represented as vectors in $[-ntB, ntB]^n$, $t$ curves represented by their $j$-invariants, and $t$ authentication paths of length $s$. The $t$ authentication paths eventually merge before the root, thus some hash values will be repeated. We can save some space by only sending the hash values once, in some standardized order: the worst case happening when no path merges before level $\log(t)$, no more than $t(s - \log(t))$ hash values need to be sent as part of the signature. In total, a signature requires at most $t\lceil n\log(2ntB + 1)\rceil + t\log(p) + t\lambda(s - \log(t))$ bits. For our parameters $t = 8$, $s = 16$ and $\lambda = 128$, this adds about 2 kilobytes to the signature of Section 4. Note that this is still an order of magnitude smaller than the best hash-based signature schemes, while being stateless, and also still of the same size as the shortest known lattice-based signatures.

Concerning security, the proofs of the previous sections, and that of [24, Appendix B] can be combined to prove the following theorem.

**Theorem 4.** *The signature scheme of this section is unforgeable under a chosen message attack under the following assumptions:*

- *Problem 3;*
- *The* multi-function multi-target second-preimage resistance *of the keyed hash function $M$;*
- *The pseudo-randomness of* $\mathsf{PRF}_{\text{secret}}$*;*

*when the hash function $H$ and the pseudo-random functions* $\mathsf{PRF}_{\text{key}}$ *and* $\mathsf{PRF}_{\text{mask}}$ *are modeled as random oracles (ideal random functions).*

Like in the previous section, it is possible to replace Problem 3 with Problem 1, modulo some additional assumptions. Both proofs are straightforward adaptations, and we omit them for conciseness. As already noted, the proofs are not tight, however the part concerned with the second-preimage resistance of $M$ is.

## 6 Performance

Table 2 gives some estimates of cost for the schemes presented in Sections 3, 4, 5. The rows of the table are divided into three sections.

The first section of the table (under the heading "Exact") reports the parameter sizes, as a number of bits, already computed in each section, where $\lambda$ is the security parameter, $n, B$ and $s$ are as described previously (in Section 3 we have $s = 1$). To simplify the expressions we assume that all hash functions have $\lambda$-bit outputs, and we set the parameter $t = \lambda/s$.

In all sections we give a rough lower bound for the performance of the keygen and sign/verify algorithms, in terms of $\mathbb{F}_p$-operations. The lower bound only takes into account the number of operations needed to compute and evaluate the isogeny path, and so the exact cost may be higher. The operation count is based on the following estimates.

1. Based on [13,35], we estimate that computing/evaluating an isogeny of degree $\ell$, when given a kernel point, costs $O(\ell)$ operations.
2. By the prime number theorem $\sum_{i=1}^{n} \ell_i \sim \frac{1}{2}n^2 \ln(n)$, and the estimate is very accurate already for $n > 3$.

Putting these estimates together, an ideal with exponent vector within $[-C, C]^n$ can be evaluated in $O(Cn^2 \log(n))$ operations on average and in the worst case. We note that the above estimate is not likely to be the dominant part in the computation, especially asymptotically, as scalar multiplications of elliptic points are likely to dominate. However, estimating this part of the algorithm is much more complex and dependent on specific optimizations, we thus leave a more precise analysis for future work.

The second section of rows in the table (under the heading "Asymptotic") gives asymptotic estimates in terms only of the security parameter $\lambda$, and the parameter of $s$ of Section 4. We now give a brief justification for the parameter restrictions in terms of $\lambda$.

1. Kuperberg's algorithm is believed to require at least $2^{\sqrt{\log(N)}}$ operations in a group of size $N$. In our case $N > \sqrt{p}$. Taking $\log(p) > 2\lambda^2$ gives

$$\sqrt{\log(N)} > \sqrt{\tfrac{1}{2}\log(p)} > \sqrt{\tfrac{1}{2}2\lambda^2} = \lambda.$$

   So we choose $\log(p) \approx 2\lambda^2$.
2. To resist a classical meet-in-the-middle attack we need $(2B + 1)^n > 2^{2\lambda}$. For security against Tani's quantum algorithm we may require $(2B + 1)^n > 2^{3\lambda}$, and so $n \log(B) \sim 3\lambda$, although the work of Adj et al. [2] suggests this may be too cautious. In any case, we have $n \log(B) = \Omega(\lambda)$.
3. Assuming that one wants to optimize for (asymptotic) performance, the best choice is then to take $B = O(1)$ and $n = \Omega(\lambda)$, which means that the prime ideals $\mathfrak{l}_i$ have norm $\ell_i = \Omega(n \log(n)) = \Omega(\lambda \log(\lambda))$. Note that this is compatible with the requirement $\log(p) > 2\lambda^2$, since $\sum_{i=1}^{n} \ln(\ell_i) \sim n \ln(n) \sim \lambda \log(\lambda)^2$.
4. Instead of measuring performance in terms $\mathbb{F}_p$-operations, here we measure them in terms of bit-operations. After substituting $B$ and $n$, this adds a factor $\lambda^2 \log(\lambda)$ in front of the lower bound if using fast (quasi-linear complexity) modular arithmetic.

Note that our asymptotic choices forbid the key space from covering the whole class group. If the conditions of Problem 1 are wanted, different choices must be made for $n$ and $B$. In this case it is best to choose primes of the form $p + 1 = 4\prod_{i=1}^{n} \ell_i$, as in CSIDH [10]. Then, $n \log(n) \sim \log(p) \sim 2\lambda^2$ and so we have $n \sim \lambda^2/\log(\lambda)$. To have a distribution of ideal classes close to uniform we need $(2B+1)^n \gg \sqrt{p}$ and so $\log(B) > \log(\sqrt{p})/n \sim \log(\lambda)$. Hence $B > \sqrt{n}$, making all asymptotic bounds considerably worse.

The third block of rows (under the heading "CSIDH") gives concrete sizes obtained by fixing $\lambda = 128$ and $s = 16$ and using the CSIDH-1 primitive, i.e., $(n, B, \log(p)) = (74, 5, 500)$. We estimate these parameters to correspond to the NIST-1 security level. Note that we are able to get smaller signatures at similar cost, for example see the various options in Table 3 (and one can also potentially consider $s > 16$, such as $(s, t) = (21, 6)$). However, for Table 2 we choose the same parameters as [10] so that we are able to refer to their running-time computations. We estimate real-world performance, using as baseline the worst-case time for one isogeny action in CSIDH. In [10], for an exponent vector in $[-B, B]^n$, this time is reported to be 0.1 seconds. Accordingly, we multiply this time by the size of the exponent vector to obtain our estimates. Note that the estimates are very rough, as they purposely ignore other factors such as hash tree computations and rejections in signing. However the results in [24,5] show that hash trees much larger than ours can be computed in a fraction of the time we need to compute isogenies. On the other hand, rejections that happen during signing multiply the running time by a small constant, typically less than 3 on average.

| | Rejection sampling (Section 3.2) | Shorter signatures (Section 4) | Smaller public keys (Section 5) |
|---|---|---|---|
| **Exact** | | | |
| Sig size | $\lambda\lceil n\log(2n\lambda B+1)\rceil+\lambda$ | $\frac{\lambda}{s}\lceil n\log(2n\frac{\lambda}{s}B+1)\rceil+\lambda$ | $\frac{\lambda}{s}(\lceil n\log(2n\frac{\lambda}{s}B+1)\rceil+\log p)+\lambda(\lambda-\frac{\lambda}{s}\log\frac{\lambda}{s})$ |
| PK size | $\log p$ | $2^s\log p$ | $2\lambda$ |
| SK size | $n\log(2B+1)$ | $\lambda$ | $(2^s+1)\lambda$ |
| Performance ($\mathbb{F}_p$-ops) | | | |
| $\rightarrow$ keygen | $\Omega\big(Bn^2\log(n)\big)$ | $\Omega\big(2^sBn^2\log(n)\big)$ | $\Omega\big(2^sBn^2\log(n)\big)$ |
| $\rightarrow$ sign/verify | $\Omega\big(\lambda^2Bn^3\log(n)\big)$ | $\Omega\big((\lambda/s)^2Bn^3\log(n)\big)$ | $\Omega\big((\lambda/s)^2Bn^3\log(n)\big)$ |
| **Asymptotic** | | | |
| Sig size | $O(\lambda^2\log(\lambda))$ | $O((\lambda^2/s)\log(\lambda))$ | $O(\lambda^3/s)$ |
| PK size | $2\lambda^2$ | $2^{s+1}\lambda^2$ | $2\lambda$ |
| SK size | $3\lambda$ | $\lambda$ | $(2^s+1)\lambda$ |
| Performance (bits) | | | |
| $\rightarrow$ keygen | $\Omega\big(\lambda^4\log(\lambda)^2\big)$ | $\Omega\big(2^s\lambda^4\log(\lambda)^2\big)$ | $\Omega\big(2^s\lambda^4\log(\lambda)^2\big)$ |
| $\rightarrow$ verify | $\Omega\big(\lambda^7\log(\lambda)^2\big)$ | $\Omega\big((\lambda^7/s^2)\log(\lambda)^2\big)$ | $\Omega\big((\lambda^7/s^2)\log(\lambda)^2\big)$ |
| **CSIDH** | | | |
| Performance (bits) | | | |
| Sig size | 19600 B | 944 B | 3092 B |
| PK size | 63 B | 4032 KB | 32 B |
| SK size | 32 B | 16 B | 1024 KB |
| Est. keygen time | 0.1 s | 6554 s | 6554 s |
| Est. sig/verify time | 123136 s | 474 s | 474 s |

**Table 2.** Parameter size and performance of the various signature protocols. Parameters taken in the asymptotic analysis are: $\log p\sim 2\lambda^2$, $n\log(B)\sim 3\lambda$, $B=O(1)$. The entry CSIDH is for parameters $(\lambda,n,B,\log_2(p))=(128,74,5,16,500)$ with $(s,t)=(1,128)$ in the first column and $(s,t)=(16,8)$ in the second two columns. All logarithms are in base 2.

| $n$ | B | $\lceil \log_2(2ntB+1) \rceil$ | Signature size (bytes) |
|---|---|---|---|
| 20 | 3275 | 20 | 416 |
| 28 | 293 | 17 | 492 |
| 33 | 124 | 16 | 544 |
| 37 | 55 | 15 | 571 |
| 46 | 22 | 14 | 660 |

**Table 3.** Parameter choices for small signatures, with $(s, t) = (16, 8)$, at around 128-bit classical security level. Signature size is $nt\lceil \log_2(2ntB+1) \rceil + 128$ bits.

## 7 Conclusions

We have given a signature scheme suitable for the CSIDH isogeny setting. This solves an unresolved problem in Stolbunov's thesis. We have also shown how to get shorter signatures by increasing the public key size. We do not know how to obtain a similar tradeoff between public key size and signature size for the schemes of Yoo et al [42] or Galbraith et al [22] based on the SIDH setting.

## Acknowledgements

Thanks to Samuel Dobson for doing some experiments with discrete Gaussians.

## References

1. Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 572–590. Springer, 2012.
2. Gora Adj, Daniel Cervantes-Vázquez, Jesús-Javier Chi-Domínguez, Alfred Menezes, and Francisco Rodríguez-Henríquez. On the cost of computing isogenies between supersingular elliptic curves. In *Selected Areas in Cryptography 2018*, Lecture Notes in Computer Science, Berlin, Heidelberg, 2018. Springer Berlin / Heidelberg. to appear.
3. László Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
4. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM, 2006.
5. Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, and Peter Schwabe. Sphincs+, November 2017.
6. Jean-François Biasse, Claus Fieker, and Michael J. Jacobson. Fast heuristic algorithms for computing relations in the class group of a quadratic order, with applications to isogeny evaluation. *LMS Journal of Computation and Mathematics*, 19(A):371–390, 2016.
7. Jean-François Biasse, Annamaria Iezzi, and Michael J. Jacobson Jr. A note on the security of CSIDH. arXiv:1806.03656, 2018.
8. Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH and ordinary isogeny-based schemes. IACR Cryptology ePrint Archive 2018/537, 2018.
9. Reinier Bröker, Denis Xavier Charles, and Kristin E. Lauter. Evaluating large degree isogenies and applications to pairing based cryptography. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 100–112. Springer, 2008.
10. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. eprint 2018/383, 2018.

11. Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.
12. Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.
13. Craig Costello and Huseyin Hisil. A simple and compact algorithm for sidh with arbitrary degree isogenies. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 303–329, Cham, 2017. Springer International Publishing.
14. Jean-Marc Couveignes. Hard homogeneous spaces. eprint 2006/291, 2006.
15. David A. Cox. *Primes of the form x2 + ny2: Fermat, class field theory, and complex multiplication*. Wiley, 1997.
16. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2013.
17. Luca De Feo. Mathematics of isogeny based cryptography. Notes from a summer school on Mathematics for Post-quantum cryptography, 2017. `http://defeo.lu/ema2017/poly.pdf`.
18. 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.
19. Luca De Feo, Jean Kieffer, and Benjamin Smith. Towards practical key exchange from ordinary isogeny graphs. eprint 2018/485, 2018.
20. Steven D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.
21. Steven D. Galbraith, Florian Hess, and Nigel P. Smart. Extending the GHS weil descent attack. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 29–44. Springer, 2002.
22. Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.
23. James L Hafner and Kevin S McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American mathematical society*, 2(4):837–850, 1989.
24. Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography – PKC 2016*, pages 387–416, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
25. David Jao and Vladimir Soukharev. A subexponential algorithm for evaluating large degree isogenies. In *ANTS*, pages 219–233, 2010.
26. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586. Springer, 2018.
27. Alexey Yuri Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.
28. Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal of Computing*, 35(1):170–188, 2005.
29. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, 2009.
30. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.
31. Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York.
32. National Institute of Standards and Technology. Announcing request for nominations for public-key post-quantum cryptographic algorithms, 2016.
33. Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Hash function requirements for Schnorr signatures. *J. Mathematical Cryptology*, 3(1):69–87, 2009.
34. Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv:quant-ph/0406151, June 2004. `http://arxiv.org/abs/quant-ph/0406151`.
35. Joost Renes. Computing isogenies between montgomery curves using the action of (0, 0). In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 229–247, Cham, 2018. Springer International Publishing.
36. Daniel Shanks. On Gauss and composition. In *Number Theory and Applications*, pages 163–204. NATO – Advanced Study Institute, Kluwer Academic Press, 1989.
37. Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *GTM*. Springer, 1986.

38. Anton Stolbunov. Cryptographic schemes based on isogenies. Doctoral thesis, NTNU, 2012.
39. Andrew Sutherland. Elliptic curves. Lecture notes from a course (18.783) at MIT, 2017. `http://math.mit.edu/classes/18.783/2017/lectures`.
40. Jean Vélu. Isogénies entre courbes elliptiques. *Comptes Rendus de l'Académie des Sciences de Paris*, 273:238–241, 1971.
41. Lawrence C. Washington. *Elliptic curves: Number theory and cryptography, 2nd ed.* CRC Press, 2008.
42. Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A post-quantum digital signature scheme based on supersingular isogenies. In *Financial Crypto*, volume 2017, 2017.

## A Variants

One can consider various ideas to get more efficient (i.e., faster signing and verification) or more compact signatures.

1. Following Stolbunov one could use higher powers for the smaller primes.
   This is definitely worth looking at. We could take $B = 1$ for some of the larger primes (e.g., the ones bigger than 50), and then use much larger values for $B$ for the primes $3, 5, 7$ etc. If $B_i$ is the bound used for $\mathfrak{l}_i$ then the full key space is $\prod_i (2B_i + 1)$.
   The main problem is that a change to a single $B_i$ makes very little difference to the key size. For example, doubling $B_1$ only adds roughly $+1$ to the logarithm of the product, which is the security parameter. Hence the main factor in having a large security parameter is using lots of distinct primes $\mathfrak{l}_i$, which automatically means a high cost for signing and verification since almost all $f_{k,i}$ will be non-zero.
2. A natural idea is to sample the exponents $e_i$ from a discrete Gaussian distribution (or perhaps some other distribution), as has been done with lattice signatures. We could hope that this leads to shorter signatures.
   Suppose the $e_i$ are sampled from a discrete Gaussian distribution with parameter $\sigma$, so that the standard deviation is close to $\sigma$. The entropy of the continuous Gaussian distribution with standard deviation $\sigma$ is $\log(2\pi e\sigma^2)/2$.
   For example, take $n = 50$, $s = 16, t = 8$ and choose $\sigma = 5$ (so almost all values $e_i$ will lie in $[-15, 15]$ but occasionally one is larger than this. Then
   $$n \log(2\pi e\sigma^2)/2 \approx 218$$
   so determining the $e_i$ using a meet-in-the-middle strategy should require at least $2^{109}$ iterations, and realistically much more than this since organising a search based on the entropy is hard to do. For post-quantum security we might want to replace $2\lambda$ with $3\lambda$ in the above.
   We following the methods and results of Lyubashevsky [30]. Lemma 4.3(3) of [30] shows we can bound the norm $\|\mathbf{e}\|$ by $T = 2\sigma\sqrt{n}$. Now we need to choose the $f_{k,i}$ from a discrete Gaussian with parameter $\sigma'$, so that the distribution of $f_{k,i} - e_i$ is close (within statistical distance, though we could probably use Renyi divergence to get better results) to the discrete Gaussian with parameter $\sigma'$. Lemma 4.6 of [30] suggests that $\sigma' = T\sqrt{\log(n)}$ is sufficient, though in practice one usually chooses $\sigma' = \alpha T$ where $\alpha \approx 10$. In our case we need to apply rejection sampling to all $t$ vectors $\mathbf{z}_k$ simultaneously, which leads to an additional factor.
   For our choices $n = 50, t = 8, \sigma = 5$ this may give $\sigma' \approx 3000$. If we use some kind of compact coding of integers distributed as Gaussians [16] then signature size would be at best $nt \log(2\pi e(\sigma')^2)/2$ bits. For our example parameters this would be between 7000-8000 bits, or around one kilobyte again.
   The best approach seems to be to sample $\mathbf{e}$ *uniformly* with coefficients in $[-B, B]$, while sampling $\mathbf{f}$ from a discrete Gaussian. Taking $n = 74, B = 5$ we have $\|\mathbf{e}| \leq T = 28$ with high probability. Taking $\sigma' = 10\sqrt{t}T \approx 790$ potentially gives signatures of around 900 bytes.
   The numbers in this section may be updated in response to further experiments and optimizations.
3. One might try to trade-off the size of exponent vectors and the rejection probability. Lemma 2 is about sampling $f_{k,i} \in [-(nt + 1)B, (nt + 1)B]$ and gives probability of acceptance $\approx 1/e \approx 0.368$. A simple modification of the proof shows that, for any $u > 0$, if one samples $f_{k,i} \in [-u(nt + 1)B, u(nt + 1)B]$ and accepts only those $\mathbf{z}_{k,i} \in [-untB, untB]$, then the acceptance probability is approximately $e^{-1/u}$.
   Taking $u = 1/2$ roughly halves the time spent on computing $\mathfrak{b}_k * E$, but changes the acceptance probability to $e^{-2} \approx 0.135$; overall this is worse than the original proposal since $2e^{-2} \approx 0.271 < e^{-1}$. Similarly, taking $u = 2$

doubles the time spent on computing $\mathfrak{b}_k * E$, but changes the acceptance probability to $e^{-1/2} \approx 0.607$; again this is worse on average than our proposal.
Indeed, if $T$ is the cost for $nt$ computations of $\mathfrak{b}_k * E$ then the expected cost of signing is $uTe^{1/u}$. Since $f(x) = xe^{1/x}$ is minimised at $x = 1$ it follows that taking $u = 1$ is the optimal choice.

## B Tight security reduction based on lossy keys

We now explain how to implement lossy keys in our setting. This allows us to use the methods of Kiltz, Lyubashevsky and Schaffner [26] (that build on work of Abdalla, Fouque, Lyubashevsky and Tibouchi [1]) to obtain signatures from lossy identification schemes. This approach gives a *tight reduction* in the *quantum random oracle model*.

Here's the basic idea to get a lossy scheme, using uniform distributions for simplicity (one can also use discrete Gaussians in this setting): Take a very large prime $p$ so that the ideal class group is very large, but use relatively small values for $n$ and $B$ so that $\{\mathfrak{a} = \prod_{i=1}^n \mathfrak{l}_i^{e_i} : |e_i| \leq B\}$ is a very small subset of the class group.[5] The real key is $(E, E_A = \mathfrak{a} * E)$ for such an $\mathfrak{a}$. The lossy key is $(E, E_A)$ where $E_A$ is a uniformly random curve in the isogeny class. Further, choose parameters so that the $f_{k,i}$ are also such that $\{\mathfrak{b} = \prod_{i=1}^n \mathfrak{l}_i^{f_{k,i}} : |f_{k,i}| \leq (nt+1)B\}$ is a small subset of the ideal class group. In the case of a real key, the signatures define ideals that correspond to "short" paths from $E$ or $E_A$ to a curve $\mathcal{E}$. In the case of a lossy key, then such ideals do not exist, as for a curve $\mathcal{E}$ it is not the case that there is a short path from $E$ to $\mathcal{E}$ AND a short path from $E_A$ to $\mathcal{E}$.

In the remainder of this section we develop these ideas.

### B.1 Background definitions

We closely follow Kiltz, Lyubashevsky and Schaffner [26]. A *canonical identification scheme* consists of algorithms $(\mathsf{IGen}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ and a set $\mathsf{ChSet}$. The randomised algorithm $\mathsf{IGen}(1^\lambda)$ outputs a key pair $(pk, sk)$. The deterministic algorithm $\mathsf{P}_1$ takes $sk$ and randomness $r_1$ and computes $(W, \mathsf{St}) = P_1(sk, r_1)$. Here $\mathsf{St}$ denotes state information to be passed to $P_2$. A challenge $c$ is sampled uniformly from $\mathsf{ChSet}$. The deterministic algorithm $P_2$ then computes $Z = P_2(sk, W, c, \mathsf{St}, r_2)$ or $\perp$, where $r_2$ is the randomness. The output $\perp$ corresponds to an abort in the "Fiat-Shamir with aborts" paradigm. We require that $\mathsf{V}(pk, W, c, Z) = 1$ for a correctly formed transcript $(W, c, Z)$.

We assume, for each value of $\lambda$, there are well-defined sets $\mathcal{W}$ and $\mathcal{Z}$, such that $\mathcal{W}$ contains all $W$ output by $\mathsf{P}_1$ and $\mathcal{Z}$ contains all $Z$ output by $\mathsf{P}_2$. The scheme is *commitment recoverable* if, given $c$ and $Z = P_2(sk, W, c, \mathsf{St})$, there is a unique $W \in \mathcal{W}$ such that $\mathsf{V}(pk, W, c, Z) = 1$ and this $W$ can be efficiently computed from $(pk, c, Z)$

A canonical identification scheme is $\epsilon_{zk}$-naHVZK *non-abort honest verifier zero knowledge* if there is a simulator that given only $pk$ outputs $(W, c, Z)$ whose distribution has statistical distance at most $\epsilon_{zk}$ from the output distribution of the real protocol conditioned on $P_2(sk, W, c, \mathsf{St}, r_2) \neq \perp$.

A *lossy identification scheme* is a canonical identification scheme as above together with a lossy key generation algorithm $\mathsf{LossIGen}$, which is a randomised algorithm that on input $1^\lambda$ outputs $pk$. An adversary against a lossy identification scheme is a randomised algorithm $A$ that takes an input $pk$ and returns 0 or 1. The advantage of an adversary against a lossy identification scheme is

$$\mathrm{Adv}^{\mathsf{LOSS}}(A) = \left|\Pr\left(A(pk) = 1 : pk \leftarrow \mathsf{LossIGen}(1^\lambda)\right) - \Pr\left(A(pk) = 1 : pk \leftarrow \mathsf{IGen}(1^\lambda)\right)\right| .$$

The two security properties of a lossy identification scheme are:

1. There is no polynomial-time adversary that has non-negligible advantage $\mathrm{Adv}^{\mathsf{LOSS}}$ in distinguishing real and lossy keys.

[5] It might even be possible to consider working with subgroups, in the quantum algorithm case where the class group structure is known.

2. The probability, over $(pk, W, c)$ where $pk$ is an output of the lossy key generation algorithm $\mathsf{LossIGen}$, $W \leftarrow \mathcal{W}$ and $c \leftarrow \mathsf{ChSet}$, that there is some $Z \in \mathcal{Z}$ with $\mathsf{V}(pk, W, c, Z) = 1$, is negligible.
   This will allow to show that no unbounded quantum adversary can pass the identification protocol (or, once we have applied Fiat-Shamir, forge a signature) with respect to a lossy public key, because with overwhelming probability no such signature exists.

## B.2 Scheme

We can re-write our scheme in this setting, see Figure 2. Here we are assuming that $E$ is a supersingular elliptic curve with $j(E) \in \mathbb{F}_p$ where $p$ satisfies the constraint

$$\sqrt{p} > (4(nt+1)B+1)^n 2^\lambda \tag{3}$$

This bound is sufficient for the keys to be lossy.

**Algorithm 4** $\mathsf{IGen}$

```
Input: $B$, $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$, $E$
Output: $sk = \mathbf{e}$ and $pk = E_A$
1: $\mathbf{e} \leftarrow [-B, B]^n$
2: $E_A = (\prod_{i=1}^n \mathfrak{l}_i^{e_i}) * E$
3: return $sk = \mathbf{e}$, $pk = E_A$
```

**Algorithm 5** $\mathsf{P}_1$

```
Input: $(E, E_A)$, $r_1$
Output: $W = (j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t))$, $\mathsf{St} = (\mathbf{f}_1, \ldots, \mathbf{f}_t)$
1: for $k = 1, \ldots, t$ do
2:     $\mathbf{f}_k \leftarrow [-(nt+1)B, (nt+1)B]^n$ using $\mathsf{PRF}(r_1)$
3:     $\mathcal{E}_k = (\prod_{i=1}^n \mathfrak{l}_i^{f_{k,i}}) * E$
4: end for
5: return $(j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t))$, $(\mathbf{f}_1, \ldots, \mathbf{f}_t)$
```

**Algorithm 6** $\mathsf{P}_2$

```
Input: $(E, E_A)$, $\mathbf{e}$, $W$, $c$, $\mathsf{St}$, $r_2$
Output: $Z = (\mathbf{z}_1, \ldots, \mathbf{z}_t)$
1: Parse $c$ as $b_1 \| \cdots \| b_t$
2: for $k = 1, \ldots, t$ do
3:     if $b_k = 0$ then
4:         $\mathbf{z}_k = \mathbf{f}_k$
5:     else
6:         $\mathbf{z}_k = \mathbf{f}_k - \mathbf{e}$
7:     end if
8:     if $\mathbf{z}_k \notin [-ntB, ntB]^n$ then
9:         return $\perp$
10:    end if
11: end for
12: return $\sigma = (\mathbf{z}_1, \ldots, \mathbf{z}_t)$
```

**Algorithm 7** $\mathsf{V}$

```
Input: $(E, E_A)$, $(W, c, Z)$
Output: Valid/Invalid
1: Parse $W$ as $(j_1, \ldots, j_t)$
2: Parse $c$ as $b_1 \| \cdots \| b_t$
3: Parse $Z$ as $(\mathbf{z}_1, \ldots, \mathbf{z}_t)$
4: for $k = 1, \ldots, t$ do
5:     if $b_k = 0$ then
6:         $\mathcal{E}_k = (\prod_{i=1}^n \mathfrak{l}_i^{z_{k,i}}) * E$
7:     else
8:         $\mathcal{E}_k = (\prod_{i=1}^n \mathfrak{l}_i^{z_{k,i}}) * E_A$
9:     end if
10: end for
11: if $(j_1, \ldots, j_t) = (j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t))$ then
12:     return Valid
13: else
14:     return Invalid
15: end if
```

**Fig. 2.** The identification protocol. Note that $P_1$ does not need $sk$, while $P_2$ does not use $r_2$ (it really is deterministic) and does not use $W$. Also note that the scheme is commitment recoverable.

Now we state the generic deterministic signature construction from Kiltz, Lyubashevsky and Schaffner [26]: The key generation and verification algorithms are the same as Figure 2. The signing algorithm is given in Figure 3. Since the identification protocol is commitment recoverable, the signatures can be shortened to be $(c, Z)$ instead of $(W, Z)$.

**Algorithm 8** Deterministic Signing algorithm

```
Input: $(pk, sk)$, $K$, $\mathsf{msg}$
Output: $\sigma$
 1: $l = 0, Z = \perp$
 2: while $Z = \perp$ and $l \leq l_0$ do
 3:     $(W, \mathsf{St}) = \mathsf{P}_1(sk, \mathsf{PRF}_K(0, \mathsf{msg}, l))$
 4:     $c = H(W, \mathsf{msg})$
 5:     $Z = \mathsf{P}_2(sk, W, c, \mathsf{St}, \mathsf{PRF}_K(1, \mathsf{msg}, l))$
 6: end while
 7: if $Z = \perp$ then
 8:     return $\perp$
 9: else
10:     return $(W, Z)$
11: end if
```

**Fig. 3.** The deterministic signature scheme of Kiltz, Lyubashevsky and Schaffner [26]. Here $K$ is a PRF key that is internal to the signing algorithm and is not required for verification.

### B.3 Proofs

We now explain that our identification scheme satisfies the required properties, from which the security of the signature scheme will follow from Theorem 3.1 of [26].

We make some heuristic assumptions.

**Heuristic 1:** There are at least $\sqrt{p}$ supersingular elliptic curves with $j$-invariant in $\mathbb{F}_p$.

This assumption, combined with the bound $\sqrt{p} \gg (4(nt+1)B)^n$ of equation (3), implies that the curves $\mathcal{E}_k$ constructed by algorithm $\mathsf{P}_1$ are a negligibly small proportion of all such curves.

**Heuristic 2:** Each choice of $\mathbf{f}_k \in [-(nt+1)B, (nt+1)B]^n$ gives a unique value for $j(\mathcal{E}_k)$.

This is extremely plausible given equation (3). It implies that the min-entropy of the values $W$ output by $\mathsf{P}_1$ is extremely high (more than sufficient for the security proofs).

Under heuristic assumption 1, we now show that the keys are lossy. The lossy key generator outputs a pair $(E, E_A)$ where $E$ and $E_A$ are randomly sampled supersingular elliptic curves with $j(E), j(E_A) \in \mathbb{F}_p$. To implement this one constructs a supersingular curve with $j$-invariant in $\mathbb{F}_p$ and then runs long pseudorandom walks in the isogeny graph until the uniform mixing bounds imply that $E_A$ is uniformly distributed.

**Lemma 5.** *Let parameters satisfy the bound of equation (3) and suppose heuristic 1 holds. Let $(E, E_A)$ be a key output by the lossy key generator. Then with overwhelming probability there is no ideal $\mathfrak{a} = \prod_{i=1}^n \mathfrak{l}_i^{f_i}$ such that $\mathbf{f} \in [-2(nt+1)B, 2(nt+1)B]^n$ and $j(E_A) = j(\mathfrak{a} * E)$.*

*Proof.* If $f_i \in [-2(nt+1)B, 2(nt+1)B]$ then there are $4(nt+1)B+1$ choices for each $f_i$ and so at most $(4(nt+1)B+1)^n$ choices for $\mathfrak{a}$. Given $E$ it means there are at most that many $j(\mathfrak{a} * E)$. Since $E_A$ is uniformly and independently sampled from a set of size at least $\sqrt{p} > (4(nt+1)B+1)^n 2^\lambda$, the probability that $j(E_A)$ lies in the set of all possible $j(\mathfrak{a} * E)$ is at most $1/2^\lambda$, which is negligible. □

We consider the following decisional problem. It would be an interesting problem to give a "search to decision" reduction in this context (showing that if one can solve Problem 4 then one can solve Problem 2). This seems to be non-trivial.

*Problem 4.* Consider two distributions on pairs $(E, E_A)$ of supersingular ellptic curves over $\mathbb{F}_p$. Let $\mathcal{D}_1$ be the output distribution of the algorithm $\mathsf{lGen}$. Let $\mathcal{D}_2$ be the uniform distribution (i.e., output distribution of the lossy key generation algorithm). The *decisional short isogeny problem* is to distinguish the two distributions when given one sample.

The next result shows the second part of the security property for lossy keys.

**Lemma 6.** *Assume heuristic 1. Let $pk$ be an output of the lossy key generation algorithm* $\mathsf{LosslGen}$. *Let $W \leftarrow \mathcal{W}$ be an output of $\mathsf{P}_1$. Let $c \leftarrow \mathsf{ChSet}$ be a uniformly chosen challenge. Then the probability that there is some $Z \in \mathcal{Z}$ with $\mathsf{V}(pk, W, c, Z) = 1$, is negligible.*

*Proof.* Let $pk = (E, E_A)$ be an output of $\mathsf{LosslGen}(1^\lambda)$. By Lemma 5 we have that with overwhelming probability $j(E_A) \neq j(\mathfrak{a} * E)$ for all ideals $\mathfrak{a}$ of the form in Lemma 5. Let $W = (j(\mathcal{E}_1), \ldots, j(\mathcal{E}_t))$ be an element of $\mathcal{W}$, so that each $\mathcal{E}_k$ is of the form $\mathfrak{a}_k * E$ where $\mathfrak{a}_k = \prod_i \mathfrak{l}_i^{f_{k,i}}$ for $f_{k,i} \in [-(nt+1)B, (nt+1)B]$.

Let $c \leftarrow \mathsf{ChSet}$ be a uniformly chosen challenge, which means that $c \neq 0$ with overwhelming probability. Then there is some $k$ with $c_k \neq 0$ and so if $Z$ was to satisfy the verification algorithm $\mathsf{V}(pk, W, c, Z) = 1$ then it would follow that $\mathbf{z}_k$ gives an ideal $\mathfrak{c}_k$ such that $j(\mathcal{E}_k) = j(\mathfrak{c}_k * E_A)$. From $\mathfrak{a}_k * E \cong \mathcal{E}_k \cong \mathfrak{c}_k * E_A$ it follows that $E_A \cong (\mathfrak{c}_k^{-1}\mathfrak{a}_k) * E$. But $\mathfrak{c}_k^{-1}\mathfrak{a}_k = \prod_i \mathfrak{l}_i^{f_{k,i} - z_{k,i}}$, which violates the claim about $E_A$ corresponding to Lemma 5. Hence $Z$ does not exist and the result is proved. $\square$

Note that Heuristic 2 also shows that there are "unique responses" in the sense of Definition 2.7 of [26] (not just computationally unique, but actually unique). But we won't need this for the result we state.

We now discuss *no-abort honest verifier zero-knowledge* (naHVZK). This is simply the requirement that there is a simulator that produces transcripts $(W, c, Z)$ that are statistically close to real transcripts output by the protocol.

**Lemma 7.** *The identification scheme (sigma protocol) of Figure 2 has no-abort honest verifier zero-knowledge.*

*Proof.* This is simple to show in our setting (due to the rejection sampling): Instead of choosing $W = (j((\prod_i \mathfrak{l}_i^{f_{1,i}}) * E) \ldots, j((\prod_i \mathfrak{l}_i^{f_{k,i}}) * E))$, then $c$, and then $Z = (\mathbf{z}_1, \ldots, \mathbf{z}_k)$ the simulator chooses $Z$ first, then $c$, and then sets, for $1 \leq k \leq t$, $j_k = j((\prod_i \mathfrak{l}_i^{z_{k,i}}) * E)$ when $c_k = 0$ and $j_k = j((\prod_i \mathfrak{l}_i^{z_{k,i}}) * E_A)$ when $c_k = 1$. Setting $W = (j_1, \ldots, j_k)$ it follows that $(W, c, Z)$ is a transcript that satisfies the verification algorithm. Further, the distribution of triples $(W, c, Z)$ is identical to the distribution from the real protocol since, for any choice of the private key, this choice of $W$ would have arisen for some choice of the original vectors $\mathbf{f}_k$. $\square$

**Theorem 5.** *Assume Heuristic 1, and the hardness of Problem 2. Then the signature scheme (applying Figure 3 to Figure 2) has UF-CMA security in the quantum random oracle model, with a tight security reduction.*

*Proof.* See Theorem 3.1 of [26]. In particular this theorem gives a precise statement of the advantage. $\square$

One can then combine this proof with the optimisations of Sections 4 and 5, to get a compact signature scheme with tight post-quantum security based on a merger of the assumptions corresponding to Problems 3 and 4.

## C Using the relation lattice

This section explains an alternative solution to the problem of representing an ideal class without leaking the private key of the signature scheme. This variant can be considered if a quantum computer is available during system setup. Essentially, this is the scheme from Stolbunov's thesis (see Section 3.1), which can be used securely once the relation lattice is known.

Let $\{\mathfrak{l}_1, \ldots, \mathfrak{l}_n\}$ be a set of $\mathcal{O}$-ideals that generates $\mathrm{Cl}(\mathcal{O})$. Define

$$L = \{(x_1, \ldots, x_n) \in \mathbb{Z}^n : \prod_{i=1}^{n} \mathfrak{l}_i^{x_i} \equiv (1)\}.$$

Then $L$ is a rank $n$ lattice with volume equal to $\# \operatorname{Cl}(\mathcal{O})$. We call this the *relation lattice*.

A basis for this lattice can be constructed in subexponential time using classical algorithms [23,6]. However, of interest to us is that a basis can be constructed in probabilistic polynomial time using quantum algorithms: define $f : \mathbb{Z}^n \to \operatorname{Cl}(\mathcal{O})$ by $f(x_1, \ldots, x_n) = \prod_{i=1}^n \mathfrak{l}_i^{x_i}$, then $f$ can be evaluated in polynomial time [36,12], and finding a basis for $L = \ker f$ is an instance of the Hidden Subgroup Problem for $\mathbb{Z}^n$, which can be solved in polynomial time using Kitaev's generalization of Shor's algorithm [27]. The classical approach is not very interesting since the underlying computational assumption is only subexponentially hard for quantum computers, but it might make sense in a certain setting. The quantum case would make sense in a post-quantum world where a quantum computer can be used to set up the system parameters for the system and then is not required for further use. It might also be possible to construct $(E, p)$ such that computing the relation lattice is efficient (e.g., constructing $E$ so that $\operatorname{Cl}(\operatorname{End}(E))$ has smooth order), but we do not consider such approaches in this paper.

For the remainder of this section we assume that the relation lattice is known. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a basis for $L$ Let $\mathcal{F} = \{\sum_{i=1}^n : u_i \mathbf{x}_i : -1/2 \leq u_i < 1/2\}$ be the centered fundamental domain of the basis of $L$. Then there is a one-to-one correspondence between $\mathcal{F} \cap \mathbb{Z}^n$ and $\operatorname{Cl}(\mathcal{O})$ by $(z_1, \ldots, z_n) \in \mathcal{F} \cap \mathbb{Z}^n \mapsto \prod_{i=1}^n \mathfrak{l}_i^{z_i}$. In practice one prefers a basis for $L$ so that all vectors in $\mathcal{F}$ have relatively short norm, which is achieved by taking the basis to be as short and close to orthogonal as possible. Hence one applies lattice basis reduction to obtain as "nice" a basis for $L$ as possible.

Note that, given a basis $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ for $L$ and a vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{Z}^n$ one can efficiently compute the unique vector in $\mathcal{F} \cap (\mathbf{z} + L)$ using the Babai rounding method [3].

Returning to Stolbunov's signature scheme, the solution to the problem is then straightforward: Given $\mathfrak{a} = \prod_{i=1}^n \mathfrak{l}_i^{e_i}$ and $\mathfrak{b}_k = \prod_{i=1}^n \mathfrak{l}_i^{f_{k,i}}$, a representation of $\mathfrak{b}_k \mathfrak{a}^{-1}$ is obtained by computing the vector $\mathbf{z}' = \mathbf{f}_k - \mathbf{e} = (f_{k,i} - e_i)$ and then using Babai rounding to get the unique vector $\mathbf{z}$ in $\mathcal{F} \cap (\mathbf{z}' + L)$. The vector $\mathbf{z}$ is sent as the response to the $k$-th challenge. Since $\mathfrak{b}_k$ is a uniformly chosen ideal class, the class $\mathfrak{b}_k \mathfrak{a}^{-1}$ is also uniformly distributed as an ideal class, and hence the vector $\mathbf{z} \in \mathcal{F} \cap \mathbb{Z}^n$ is uniformly distributed and carries no information about the private key.

**Lemma 8.** *If $\mathfrak{b}_k$ is a uniformly chosen ideal class then the vector $\mathbf{z} \in \mathcal{F} \cap \mathbb{Z}^n$ corresponding to $\mathbf{f}_k - \mathbf{e}$ is uniformly distributed.*

*Proof.* For fixed $\mathbf{e}$ the vector $\mathbf{z}$ depends only on the ideal class of $\mathfrak{b}_k$. But $\mathfrak{b}_k$ is uniform and independent of $\mathbf{e}$ and not known to verifier. □

If the basis for $L$ is sufficiently nice then one can obtain good bounds on the size of the vectors $\mathbf{z}$; for example some details are given in [8].

One final remark: In the security proof we need to be able to simulate the signing oracle, and hence we need to produce uniformly chosen vectors $\mathbf{z} \in \mathcal{F} \cap \mathbb{Z}^n$. The simplest way to do this is to uniformly sample $\mathbf{z}'$ in a large box in $\mathbb{Z}^n$ and then apply Babai rounding as above. It is an open problem to obtain rigorous results about the uniform distribution of ideal classes in this setting.

**Lemma 9.** *Let $B \in \mathbb{N}$. Let $\mathcal{D}_1$ be the distribution on ideal classes obtained by computing $\prod_{i=1}^n \mathfrak{l}_i^{x_i}$ over uniformly sampled $x_i \in [-B, B]$. Suppose the statistical distance between $\mathcal{D}_1$ and the uniform distribution on $\operatorname{Cl}(\mathcal{O})$ is bounded by $\epsilon$. Let $\mathcal{D}_2$ be the distribution on $\mathcal{F} \cap \mathbb{Z}^n$ defined by uniformly sampling vectors $\mathbf{x} \in [-B, B]^n$ and applying Babai rounding. Let $U$ be the uniform distribution on $\mathcal{F} \cap \mathbb{Z}^n$. Then the statistical distance between $\mathcal{D}_2$ and $U$ is at most $\epsilon$.*

*Proof.* The distribution of ideal classes $\prod_{i=1}^n \mathfrak{l}_i^{x_i}$ in $\operatorname{Cl}(\mathcal{O})$ is the same as the distribution of vectors in $\mathcal{F} \cap \mathbb{Z}^n$ obtained by Babai rounding of $\mathbf{x} = (x_1, \ldots, x_n)$. Hence, if one distribution is close to uniform then so is the other. □

One can then prove a variant of Theorem 1 and all the theorems in the paper. This approach should give rise to much smaller signatures – close to optimal size given the subexponential security of the class group action problem. But we do not consider this topic further in this paper as it requires a quantum computer during key generation.