SeaSign: Compact isogeny signatures from class group actions

Luca De Feo[1][0000−0002−9321−0773] and Steven D. Galbraith[2][0000−0001−7114−8377]

[1] Université Paris-Saclay – UVSQ, LMV, UMR CNRS 8100, Versailles, FR. https://defeo.lu/
[2] Mathematics Department, University of Auckland, NZ. s.galbraith@auckland.ac.nz

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 [51] 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 [13] (due to Castryck, Lange, Martindale, Panny and Renes) and the scheme by De Feo, Kieffer and Smith [22], 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 [41] 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 [55]) state signatures of over 100 KiB and signing/verification that take a few seconds on a PC. This can be reduced using some optimisations. For example [29] state approximately 12 KiB for this signature scheme (for classical 128-bit security level) and approximately 11 KiB for their main scheme. 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 (the order of minutes), but we hope that future work (see for example [23]) will lead to more efficient schemes.

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 [13] we propose a signature scheme whose public key is 4 MiB, signature size is 978 bytes, and verification time is under 3 minutes (signing time is three times longer than this on average, 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 3 KiB 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. This optimisation also leads to faster signing and verification. Section 5 shows how to retain the benefit of shorter signatures, while also having a short public key, by using modified Merkle trees. Section 7 sketches some optimisations and variants. Section 8 shows 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 [37] (and this security enhancement involves no increase in signature size, though the primes are larger so computations will be somewhat slower). This is the first time that lossy keys have been used in the isogeny setting. Section 9 explains that, if a quantum computer is available during parameter generation, then a much more practical signature scheme can be obtained by following the methods in Stolbunov's thesis.

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 finite set X; $\log$ denotes the logarithm in base 2; KiB and MiB denote kilobytes and megabytes respectively; 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 [50], Washington [54], Galbraith [27], Sutherland [52] and De Feo [20]. A good reference for ideal class groups and class group actions is Cox [19].

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 [53] gives an algorithm to compute an equation for E' and rational functions that enable to compute ϕ. 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 [13] 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. The principal ideal $(1) = \mathcal{O}$ is the identity element of the ideal class group. 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 homogeneous space for $\mathrm{Cl}(\mathcal{O})$. Good references for the details are Couveignes [18] and Stolbunov [51]. 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 [28] and developed further in [12,35,8]. The state of the art is [8] which computes $\mathfrak{a} * E$ 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 $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ are split prime $\mathcal{O}$-ideals of small norm ℓ_i and where $(e_1, \ldots, e_n)$ is an appropriately chosen vector of exponents. Then, the action of $\mathfrak{a}$ can be computed as a composition of isogenies of degree ℓ_i. Throughout the paper we assume that $\{\mathfrak{l}_1, \ldots, \mathfrak{l}_n\}$ 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 to invert the action of an ideal. Couveignes called Problem 1 "vectorisation" and Stolbunov called it "Group Action Inverse Problem (GAIP)". The CSIDH paper speaks of hard homogeneous spaces and calls the problem "Key recovery".

Problem 1. Given two elliptic curves E and E_A tover 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 [15] 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 [39] and Regev [47]. It is still an active area of research to assess the exact quantum hardness of these problems; see the recent papers by Biasse-Iezzi-Jacobson [9], Bonnetain-Schrottenloher [11], Jao-LeGrow-Leonardi-Ruiz-Lopez [34] and Bernstein-Lange-Martindale-Panny [7]. 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 λ 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 [33][3]; 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 ℓ_i as efficient as possible. This is done both by choosing the primes ℓ_i to be as small as possible, and also by arranging that the kernel subgroups $E[\ell_i]$ are defined over as small a field extension as possible (so that Vélu's formulas can be used). In the ordinary case, the best technique currently available to select parameters is due to De Feo, Kieffer and Smith [22]. Despite the optimisations described in [22], this technique requires years of CPU time to construct a good curve. Like [22], CSIDH [13] chooses a special prime of the form $p + 1 = 4\prod_{i=1}^{n} \ell_i$, but, instead of ordinary curves, it uses supersingular curves defined over $\mathbb{F}_p$. This makes the search for a suitable curve virtually instantaneous, and produces very efficient parameters; indeed note that the formula for $p+1$ implies that each prime ℓ_i splits in $\mathbb{Q}(\sqrt{-p})$ as a product $(\ell_i) = \mathfrak{l}_i\bar{\mathfrak{l}}_i$ of distinct prime ideals. For key exchange, CSIDH samples the exponent vectors $\mathbf{e} = (e_1, \dots, e_n) \in [-B, B]^n \subseteq \mathbb{Z}^n$ 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)

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

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 λ 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 [21] for details). This means we might need to take $(2B + 1)^n > 2^{3\lambda}$ to have post-quantum security. However, recent analyses [2,36] question the pertinence of the complexity models of the meet-in-the-middle and Tani algorithms, and advocate for more relaxed bounds.

Choosing the best values of B, n, p for large choices of λ (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.

[3] On the other hand, SIDH has the advantage that no subexponential-time algorithm is known to break it.

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 [45], as summarised 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-512 computes one class group action in 40ms on a 3.5GHz processor.

	n	$\lfloor \log_2 p \rfloor$	B	NIST level	classical security	quantum security	message size	private key size
CSIDH-512	74	510	5	1	127 bits	62 qbits	64B	37B
CSIDH-1024	130	1019	8	3	257 bits	94 qbits	127B	82B
CSIDH-1792	208	1786	10	5	449 bits	129 qbits	223B	130B

Table 1. Proposed parameters for CSIDH [13]. Effective parameters p, n and B for CSIDH-1024 and CSIDH-1792 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 more general primes than considered in CSIDH [13]. 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 8.1 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 ChSet. The randomised algorithm $\mathsf{KeyGen}(1^\lambda)$ outputs a key pair (pk, sk). The deterministic algorithm P_1 takes sk and randomness r_1 and computes $(W, \mathsf{st}) = P_1(sk, r_1)$. Here st denotes state information to be passed to P_2. A challenge c is sampled uniformly from 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 λ is a security parameter. The randomised algorithm Sign takes input the private key sk and a message 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 KeyGen to get (pk, sk) and runs $A(pk)$. The adversary A sends messages 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 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 Section 8.

For simplicity, we describe our schemes in the setting of a general class group action on a set of j-invariants of elliptic curves. In Section 3.3 we explain one small subtlety that arises when implementing the scheme in the setting of CSIDH.

3.1 Stolbunov's scheme

Section 2.B of Stolbunov's PhD thesis [51] 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 [18] 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 [51] 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 $\mathbf{f}_k - \mathbf{e} = (f_{k,i} - e_i)_{1 \leq i \leq n}$ would not be secure as it would leak the private key. Instead, Stolbunov (and also Couveignes) work in the setting where the relation lattice in the ideal class group is known; we discuss this further in Section 9. A main contribution of our paper is to give solutions to this problem (using Fiat-Shamir with aborts) that do not require to know the relation lattice.

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 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). However the cost to compute the relations in the ideal class group on a classical computer is in essentially the same asymptotic complexity class as the cost to break the scheme on a quantum computer (using the Kuperberg or Regev algorithms). Hence it may not make sense to require the Key Generation algorithm of the scheme to compute the relations in the ideal class group. On the other hand, in the fully post-quantum setting where quantum computers are readily available then the relation lattice can be computed in polynomial time. We revisit this issue in Section 9.

3.2 Using rejection sampling

To prevent signatures from leaking the private key, we use rejection sampling in exactly the way proposed by Lyubashevsky [41] 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 Section 8 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.[4] 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 $1 \leq i \leq n$. 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 $|z_{k,i}| \leq ntB$. 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

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

Hence, the probability that all of the values $z_{k,i}$ over $1 \leq k \leq t, 1 \leq i \leq n$ lie in $[-ntB, ntB]$ is at least $(1 - 1/(nt+1))^{nt}$. Using the inequality $1 - 1/(x+1) \geq e^{-1/x}$ for $x \geq 1$ 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 λ 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-512, we obtain signatures of around 20 KiB (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 $1 \leq k \leq t$, which means that for each k and each prime $\mathfrak{l}_i$ one needs to compute up to ntB isogenies of degree ℓ_i. Hence, the total number of

[4] In the scheme and analysis we apply rejection sampling to the case $b_k = 0$ as well as the case $b_k = 1$. An alternative would be to only apply rejection sampling in the case $b_k = 1$. It doesn't really matter one way or the other, since in both settings we are able to simulate a signer in the random oracle model and so the security proof works.

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.

isogeny computations is upper bounded by $(nt)^2 B$. The quadratic dependence on nt is a major inconvenience. 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 example, when $(n, t, B) = (74, 128, 5)$ then the true acceptance probability is approximately 0.36790, while $e^{-1} \approx 0.36788$.

We discuss some possible optimisations in Section 7, including the idea to use discrete Gaussians instead of uniform distributions for the vectors.

3.3 CSIDH implementation

The above description represents the isomorphism class of $\mathfrak{a} * E$ using a j-invariant. But, as explained in [24,13], in the case of supersingular curves over $\mathbb{F}_p$ there are two isomorphism classes for each j-invariant and so the j-invariant alone is not an adequate representation for $\mathfrak{a} * E$. Castryck *et al.* [13] observe that the Montgomery model for these curves provides an elegant solution to the dilemma. Instead of representing $\mathfrak{a} * E$ with a j-invariant one uses the "A coefficient" of the Montgomery equation. This works when choosing $p \equiv 3 \pmod{8}$ and using curves whose endomorphism ring is on the "floor" of the 2-isogeny volcano; we refer to Proposition 8 of [13] for the details.

In short, when implementing our signature schemes using CSIDH one should choose $p \equiv 3 \pmod{8}$ and replace the words "j-invariant" by "Montgomery coefficient". In terms of the security analysis, strictly speaking the security proofs use variants of the computational problems expressed in terms of Montgomery coefficients. It is a simple exercise to show that these problems are equivalent to problems expressed using j-invariants. Hence the theorem statements in our paper are all correct in the setting of CSIDH.

3.4 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 Section 8.

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.[5] 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 λ 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 $\mathfrak{a}$.

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

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 randomised algorithm that takes as input $h_1, \dots, h_Q \in \{0,1\}^t$ and outputs (J, σ) 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, σ) 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', σ'). 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.*

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 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 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 ϵ then the simulator solves the computational problem with probability proportional to ϵ^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 [46].

4 Smaller signatures and faster signing/verification

The signature size of the basic scheme is rather large (around 20 KiB), 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 (several megabytes). 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. This optimisation also dramatically speeds up signing and verification. In the next section we explain how to shorten the public keys again.

The basic idea is to have public keys $(E_{A,0} = \mathfrak{a}_0 * E, \ldots, E_{A,2^s-1} = \mathfrak{a}_{2^s-1} * E)$. 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 $b_1, \ldots, b_t \in \{0, 1, \ldots, 2^s - 1\}$. 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 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 1 KiB (see Table 3). Of course the public key now includes 2^{16} j-invariants (elements of $\mathbb{F}_p$) which would be around 4 MiB, 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.* [55] or Galbraith *et al.* [29].

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 $\{E_{A,0}, \ldots, E_{A,2^s-1}\}$, 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 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 based cryptography 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 ϵ, 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 $\mathfrak{b}_0, \ldots, \mathfrak{b}_{2^s-1}$ (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 π on $\{0, 1, \ldots, 2^s - 1\}$ 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 ℓ_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 ϵ. 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. □

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 Section 8 based on a less standard assumption (see Problem 4). 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 $0 \leq m < 2^s$ 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 [43] 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 [32] and SPHINCS+ [5], whose security relies on the second-preimage resistance of a keyed hash function.

Let λ 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 λ 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 $\mathbf{e}_m = \mathsf{PRF}_{\mathrm{secret}}(\mathsf{SK.seed}, m)$, and the public curves $E_{A,m} = \mathfrak{a}_m * E$, for $0 \leq m < 2^s$.

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

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

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

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

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 [32, Appendix B] shows that having M output λ-bit hashes gives a (classical) security of approximately 2^λ. See [32,5] for more details.

Typically, in hash-based signatures the secret key would only contain $\mathsf{SK.seed}$, 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 MiB 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 $\mathsf{PRF}_{\mathrm{secret}}$, 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 standardised 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 KiB to the signature of Section 4. Note that this is still less than half the size of the best stateless hash-based signature schemes (the NIST candidate SPHINCS+ [6,5] has size-optimized signatures of 8080 bytes at the NIST security level 1), and is comparable in size to stateful hash-based signatures (e.g., the IETF draft XMSS [31, § 5.3.1]) and to the shortest known lattice-based signatures.

Concerning security, the proofs of the previous sections, and that of [32, 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 is hard;*
- *The* multi-function multi-target second-preimage resistance *of the keyed hash function M;*
- *The pseudo-randomness of* $\mathsf{PRF}_{\mathrm{secret}}$*;*

when the hash function H and the pseudo-random functions $\mathsf{PRF}_{\mathrm{key}}$ *and* $\mathsf{PRF}_{\mathrm{mask}}$ *are modelled 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 λ 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 λ-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.

	Rejection sampling (Section 3.2)	Shorter signatures (Section 4)	Smaller public keys (Section 5)
Exact			
Sig size	$\lambda n\lceil\log(2n\lambda B+1)\rceil+\lambda$	$\frac{\lambda}{s}n\lceil\log(2n\frac{\lambda}{s}B+1)\rceil+\lambda$	$\frac{\lambda}{s}(n\lceil\log(2n\frac{\lambda}{s}B+1)\rceil+\log p)+\lambda(\lambda-\frac{\lambda}{s}\log\frac{\lambda}{s})$
PK size	$\log p$	$2^s\log p$	2λ
SK size	$n\log(2B+1)$	λ	$(2^s+1)\lambda$
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	$2\lambda^2$	$2^{s+1}\lambda^2$	2λ
SK size	3λ	λ	$(2^s+1)\lambda$
Performance (bits)			
$\rightarrow$ keygen	$\Omega\big(\lambda^4\log(\lambda)^2\big)$	$\Omega\big(2^s\lambda^4\log(\lambda)^2\big)$	$\Omega\big(2^s\lambda^4\log(\lambda)^2\big)$
$\rightarrow$ sign/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			
Sig size	20144 B	978 B	3136 B
PK size	64 B	4096 KiB	32 B
SK size	32 B	16 B	1024 KiB
Est. keygen time	0.03 s	1966 s	1966 s
Est. verify time	36372 s	142 s	142 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 $(\lambda,n,B,\log(p))=(128,74,5,510)$ 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. Signing time is on average 3 times the estimated verification time.

The operation count is based on the following estimates.

1. Based on [17,48], we estimate that computing/evaluating an isogeny of degree ℓ, 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 optimisations, 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 λ, and the parameter of s of Section 4. We now give a brief justification for the parameter restrictions in terms of λ.

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 $(2B + 1)^n > 2^{2\lambda}$, although the work of Adj *et al.* [2] suggests this may be too cautious. For security against Tani's quantum algorithm we may require $(2B+1)^n > 2^{3\lambda}$, and so $n \log(B) \sim 3\lambda$, though again this may not be necessary [36]. In any case, we have $n \log(B) = \Omega(\lambda)$.
3. Assuming that one wants to optimise 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 of $\mathbb{F}_p$-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 [13]. 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-512 primitive, i.e., $(n, B, \log(p)) = (74, 5, 510)$. 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 [13] 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 [13,44], for an exponent vector in $[-B, B]^n$, this time is reported to be around 30 milliseconds. 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. However the results in [32,5] show that hash trees much larger than ours can be computed in a fraction of the time we need to compute isogenies.

7 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, but it is unclear that this provides any benefit in practice.

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.

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 σ, so that the standard deviation is close to σ. The entropy of the continuous Gaussian distribution with standard deviation σ 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 2λ with 3λ in the above.

   We follow the methods and results of Lyubashevsky [42]. Lemma 4.3(3) of [42] 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 σ', 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 σ'. Lemma 4.6 of [42] 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 gives $\sigma' \approx 3500$. If we use some kind of compact coding of integers distributed as Gaussians [25] then signature size would be at best $nt\log_2(2\pi e(\sigma')^2)/2$ bits. For our example parameters this would be between 7000-8000 bits, or around 1 KiB 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 $\|\mathbf{e}| \leq T = 24$ with reasonable probability. Taking $\sigma' = 10\sqrt{t}T \approx 790$ potentially gives signatures of around 800-900 bytes.

   In conclusion, the use of discrete Gaussians does not seem to lead to shorter or faster signatures than using uniform distributions. Hence we suggest to use uniform distributions since the implementation of rejection sampling is much simpler and less error-prone.

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.

8 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 [37] (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.[6] 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.

8.1 Background definitions

We closely follow Kiltz, Lyubashevsky and Schaffner [37]. A *canonical identification scheme* consists of algorithms $(\mathsf{IGen}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ and a set ChSet. The randomised algorithm $\mathsf{IGen}(1^\lambda)$ outputs a key pair (pk, sk). The deterministic algorithm P_1 takes sk and randomness r_1 and computes $(W, \mathsf{st}) = \mathsf{P}_1(sk, r_1)$. Here st denotes state information to be passed to P_2. A challenge c is sampled uniformly from ChSet. The deterministic algorithm P_2 then computes $Z = \mathsf{P}_2(sk, W, c, \mathsf{st}, r_2)$ 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 λ, there are well-defined sets $\mathcal{W}$ and $\mathcal{Z}$, such that $\mathcal{W}$ contains all W output by P_1 and $\mathcal{Z}$ contains all Z output by P_2. The scheme is *commitment recoverable* if, given c and $Z = \mathsf{P}_2(sk, W, c, \mathsf{st})$, 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 ϵ_{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 ϵ_{zk} from the output distribution of the real protocol conditioned on $\mathsf{P}_2(sk, W, c, \mathsf{st}, r_2) \neq \perp$.

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^λ 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.
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.

8.2 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

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

[6] It might even be possible to consider working with subgroups, in the quantum algorithm case where the class group structure is known. For example, private keys could be sampled from a large subgroup and lossy keys from a non-trivial coset.

This bound is sufficient for the keys to be lossy.

Algorithm 4 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 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 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 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 [37]: 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).

8.3 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 [37].

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 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 P_1 is extremely high (more than sufficient for the security proofs).

Algorithm 8 Deterministic Signing algorithm

```
Input: $(pk, sk)$, $K$, 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 [37]. Here K is a PRF key that is internal to the signing algorithm and is not required for verification.

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 is an open challenge 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 elliptic curves over $\mathbb{F}_p$. Let $\mathcal{D}_1$ be the output distribution of the algorithm IGen. 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* $\mathsf{LossIGen}$. *Let $W \leftarrow \mathcal{W}$ be an output of 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 $\mathsf{LossIGen}(1^\lambda)$. 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 with overwhelming probability Z does not exist, and the result is proved. □

Note that Heuristic 2 also shows that there are "unique responses" in the sense of Definition 2.7 of [37] (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 $\mathbf{f}_k$. □

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

Proof. See Theorem 3.1 of [37]. In particular this theorem gives a precise statement of the advantage. □

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.

9 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. Note that this section is about signatures that involve sampling ideal classes uniformly and so the techniques can't be used in the lossy keys setting.

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

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

Then L is a rank n lattice with volume equal to $\#\,\mathrm{Cl}(\mathcal{O})$. Indeed, we have the exact sequence of Abelian groups

$$0 \to L \to \mathbb{Z}^n \to \mathrm{Cl}(\mathcal{O}) \to 1$$

where the map $f : \mathbb{Z}^n \to \mathrm{Cl}(\mathcal{O})$ is the group homomorphism $(x_1, \ldots, x_n) \mapsto \prod_i \mathfrak{l}_i^{x_i}$. We call L the *relation lattice*.

A basis for this lattice can be constructed in subexponential time using classical algorithms [30,8]. However, of interest to us is that a basis can be constructed in probabilistic polynomial time using quantum algorithms: the function $f : \mathbb{Z}^n \to \mathrm{Cl}(\mathcal{O})$ defined in the previous paragraph can be evaluated in polynomial time [49,16], 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 generalisation of Shor's algorithm [38]. 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 $\mathrm{Cl}(\mathrm{End}(E))$ 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 $\{\mathbf{x}_1,\dots,\mathbf{x}_n\}$ be a basis for L. Let $\mathcal{F} = \{\sum_{i=1}^n : u_i\mathbf{x}_i : -1/2 \le u_i < 1/2\}$ be the centred fundamental domain of the basis of L. Then there is a one-to-one correspondence between $\mathcal{F}\cap\mathbb{Z}^n$ and $\mathrm{Cl}(\mathcal{O})$ by $(z_1,\dots,z_n)\in\mathcal{F}\cap\mathbb{Z}^n \mapsto \prod_{i=1}^n \mathfrak{l}_i^{z_i}$.

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 $\mathbf{z}' = \mathbf{f}_k - \mathbf{e}$ 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. □

The above discussion fixes a particular fundamental domain and uses Babai rounding to compute an element in it, but this may not lead to the most efficient signature scheme. One can consider different fundamental domains and different "reduction" algorithms to compute $\mathbf{z}$. Since the cost of signature verification depends on the size of the entries in $\mathbf{z}$, a natural computational problem is to efficiently compute a short vector $(z_1,\dots,z_n)$ corresponding to a given ideal class; we discuss this problem in the next subsection.

9.1 Solving close vector problems in the relation lattice

Let $\mathbf{w} = (w_1,\dots,w_n)\in\mathbb{Z}^n$ be given and suppose we want to compute the isogeny $\mathfrak{a}*E$ where $\mathfrak{a} = \prod_{i=1}^n \mathfrak{l}_i^{w_i}$. Since the computation of the isogeny depends on the sizes of $|w_i|$ it is natural to first compute a short vector $(z_1,\dots,z_n)$ that represents the same element of $\mathbb{Z}^n/L$. This can be done by solving a close vector problem in the lattice L. Namely, if $\mathbf{v}\in L$ is such that $\|\mathbf{w}-\mathbf{v}\|$ is short, then $\mathbf{z} = \mathbf{w}-\mathbf{v}$ is a short vector that can be used to compute $\mathfrak{a}*E$. Hence, the problem of interest is the close vector problem in the relation lattice.

Note that most literature and algorithms for solving close lattice vector problems are with respect to the Euclidean norm, whereas for isogeny problems the natural norms are the 1-norm $\|\mathbf{z}\|_1 = \sum_{i=1}^n |z_i|$ or the ∞-norm $\|\mathbf{z}\|_\infty = \max_i |z_i|$. The choice of norm depends on how the isogeny is computed. The algorithm for computing $\mathfrak{a}*E$ given in [13] depends mostly on the ∞-norm, since the Vélu formulae are used and a block of isogenies are handled together in each iteration. However, the intuitive cost of the isogeny (and this is appropriate when using modular polynomials to compute the isogenies) is given by the 1-norm. If the entries z_i are uniformly distributed in $[-\|\mathbf{z}\|_\infty, \|\mathbf{z}\|_\infty]$ then we have $\|\mathbf{z}\|_\infty \approx \sqrt{3/n}\|\mathbf{z}\|_2$ and $\|\mathbf{z}\|_1 \approx \frac{n}{2}\|\mathbf{z}\|_\infty \approx \sqrt{3n/4}\|\mathbf{z}\|_2$.

There are many approaches to solving the close vector problem. All methods start with pre-processing the lattice using some basis reduction, and in our case one can perform a major precomputation to produce a basis customised for solving close vector problems. Once the instance $\mathbf{w}$ is provided one can perform one of the following three approaches: the Babai nearest plane method (or an iterative version of it, as done by Lindner and Peikert [40]); enumeration; reducing to SVP (the Kannan embedding technique) and running a basis reduction algorithm. The choice of method depends on the quality of the original basis, the amount of time available to spend on solving CVP (note that a reduction in the sizes of the $|z_i|$ pays dividends in the time to compute $\mathfrak{a}*E$, and so it may be worth to devote more than a few cycles to this problem).

For this paper we focus on the Babai nearest plane algorithm. Let $\mathbf{b}_1,\dots,\mathbf{b}_n$ be the (ordered) reduced lattice basis and $\mathbf{b}_1^*,\dots,\mathbf{b}_n^*$ the Gram-Schmidt vectors. Equation (4.3) of Babai [3] shows that the nearest plane algorithm on input $\mathbf{w}$ outputs a vector $\mathbf{v}\in L$ with

$$\|\mathbf{w}-\mathbf{v}\|_2^2 \le (\|\mathbf{b}_1^*\|_2^2 + \|\mathbf{b}_2^*\|_2^2 + \cdots + \|\mathbf{b}_n^*\|_2^2)/4. \tag{4}$$

Bounds on $\|\mathbf{b}_i^*\|$ are regularly discussed in the literature. For example, much work on the BKZ algorithm is devoted to understanding the sizes of these vectors; see Gama-Nguyen and Chen-Nguyen [14].

Fukase and Kashiwabara [26] have discussed lattice reduction algorithms that produce a basis that minimises the right hand size of equation (4) and hence are good for solving CVP using the nearest-plane algorithm. Blömer [10] has given a variant of the near-plane algorithm that efficiently solves CVP when given a dual-HKZ-reduced basis.

For our calculations we simply consider a BKZ-reduced lattice basis and, following Chen-Nguyen [14], assume that

$$\|\mathbf{b}_i^*\|_2 \approx \|\mathbf{b}_1\|_2^{1-0.0263(i-1)}.$$

Some similar calculations are given in [11].

9.2 Optimal signature size

We now use an idea that is implicit in the work of Couveignes [18] and Stolbunov [51] that gives signatures of optimal size when the relation lattice is known. Suppose the ideal class group is cyclic of order N and let $\mathfrak{g}$ be a generator (whose factorisation over $(\mathfrak{l}_1, \dots, \mathfrak{l}_n)$ is known). Then one can choose the private key by uniformly sampling an integer $0 \leq x < N$ and letting $\mathfrak{a} = \mathfrak{g}^x$ in $\mathrm{Cl}(\mathcal{O})$. The public key is $E_A = \mathfrak{a} * E$ as before (this computation requires "smoothing" the ideal class using the relation lattice). When signing one chooses the t random ideals $\mathfrak{b}_k$ by choosing uniform integers y_k in $[0, N)$ and computing $\mathfrak{b}_k = \mathfrak{g}^{y_k}$. As before $\mathcal{E}_k = \mathfrak{b}_k * E$. Finally, in the scheme, when $b_k = 0$ we return y_k and when $b_k = 1$ we return $y_k - x \pmod{N}$. The verifier just sees a uniformly distributed integer modulo N, and uses this to recompute $\mathcal{E}_k$ from either E or E_A (again, this requires reducing a vector modulo the relation lattice and then computing the corresponding isogenies). This scheme is clearly optimal from the point of view of signature size, since one cannot represent a random element of a group of order N in fewer than $\log_2(N)$ bits.

The method used to compute the isogenies during verification is left for the verifier to decide. In practice all users will work with the same prime p (e.g., the 512-bit CSIDH prime) in which case the relation lattice can be precomputed and optimised. The verifier then solves the CVP instances using their preferred method and then computes the isogenies.

One can then prove a variant of Theorem 1 and adapt the optimisations of Sections 4 and 5. This approach gives rise to smaller signatures, and much faster signature and verification times: in terms of the security parameter λ the size gain is logarithmic, while the performance gain is quadratic. Public/private key sizes and key generation time are unaffected. We summarise the obtained parameters and expected performance in Table 4, using the same conventions as in Table 2; we only report data on signature size and sign/verify performance, and we refer to Table 2 for the other (unchanged) entries.

	Stolbunov scheme	Shorter signatures	Smaller public keys
Exact			
Sig size	$\lambda\lceil n\log(2B+1)\rceil + \lambda$	$\frac{\lambda}{s}\lceil \log(p)/2\rceil + \lambda$	$\frac{\lambda}{s}(\lceil n\log(2B+1)\rceil + \log p) + \lambda(\lambda - \frac{\lambda}{s}\log\frac{\lambda}{s})$
Performance ($\mathbb{F}_p$-ops)	$\Omega\big(\lambda B n^2 \log(n)\big)$	$\Omega\big((\lambda/s) B n^2 \log(n)\big)$	$\Omega\big((\lambda/s) B n^2 \log(n)\big)$
Asymptotic			
Sig size	$O(\lambda^2)$	$O(\lambda^2/s)$	$O(\lambda^3/s)$
Performance (bits)	$\Omega\big(\lambda^5 \log(\lambda)^2\big)$	$\Omega\big((\lambda^5/s)\log(\lambda)^2\big)$	$\Omega\big((\lambda^5/s)\log(\lambda)^2\big)$
CSIDH			
Sig size	5888 B	271 B	2429 B
Est. verify time	13 s	2 s	2 s

Table 4. Signature size and sign/verify performance of Stolbunov's protocol, when combined with the variants of Sections 4 and 5. The conventions are the same as in Table 2: $\log p \sim 2\lambda^2$, $n\log(B) \sim 3\lambda$, $B = O(1)$ for asymptotic analysis, $(\lambda, n, \log(p)) = (128, 74, 510)$ with $(s,t) = (1,128)$ or $(16,8)$ for the CSIDH entry. All logarithms are in base 2.

10 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 trade-off between public key size and signature size for the schemes of Yoo *et al.* [55] or Galbraith *et al.* [29] based on the SIDH setting.

Acknowledgements

Thanks to Samuel Dobson for doing some experiments with discrete Gaussians. Thanks to Lorenz Panny for comments and suggestions. Thanks to Damien Stehlé for references about solving close vector problems. Thanks to the Eurocrypt referees for their careful reading of the paper.

Luca De Feo was supported by the French *Programme d'Investissements d'Avenir* under the national project RISQ n^o P141580-3069086/DOS0044212.

References

1. Abdalla, M., Fouque, P., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 572–590. Springer (2012)
2. Adj, G., Cervantes-Vázquez, D., Chi-Domínguez, J.J., Menezes, A., Rodríguez-Henríquez, F.: On the cost of computing isogenies between supersingular elliptic curves. In: Cid, C., Jr., M.J.J. (eds.) Selected Areas in Cryptography 2018. Lecture Notes in Computer Science, vol. 11349. Springer (2019)
3. Babai, L.: On Lovász lattice reduction and the nearest lattice point problem. Combinatorica **6**(1), 1–13 (1986)
4. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM CCS 2006. pp. 390–399. ACM (2006)
5. Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P.: SPHINCS+ (Nov 2017), https://sphincs.org/data/sphincs+-submission-nist.zip
6. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: SPHINCS: practical stateless hash-based signatures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 368–397. Springer (2015)
7. Bernstein, D.J., Lange, T., Martindale, C., Panny, L.: Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. to appear at EuroCrypt 2019 (2019)
8. Biasse, J., Fieker, C., Jacobson, M.J.: 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)
9. Biasse, J., Iezzi, A., Jr., M.J.J.: A note on the security of CSIDH. In: Chakraborty, D., Iwata, T. (eds.) INDOCRYPT 2018. vol. 11356, pp. 153–168. Springer (2018)
10. Blömer, J.: Closest vectors, successive minima, and dual HKZ-bases of lattices. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. Lecture Notes in Computer Science, vol. 1853, pp. 248–259. Springer (2000)
11. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH and ordinary isogeny-based schemes. IACR Cryptology ePrint Archive 2018/537 (2018)
12. Bröker, R., Charles, D.X., Lauter, K.E.: Evaluating large degree isogenies and applications to pairing based cryptography. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. Lecture Notes in Computer Science, vol. 5209, pp. 100–112. Springer (2008)
13. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S.D. (eds.) ASIACRYPT 2018. Lecture Notes in Computer Science, vol. 11274, pp. 395–427. Springer (2018)
14. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 1–20. Springer (2011)
15. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. Journal of Mathematical Cryptology **8**(1), 1–29 (2014)
16. Cohen, H.: A Course in Computational Algebraic Number Theory. Springer-Verlag New York, Inc., New York, NY, USA (1993)
17. Costello, C., Hisil, H.: A simple and compact algorithm for sidh with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 303–329. Springer International Publishing, Cham (2017)
18. Couveignes, J.M.: Hard homogeneous spaces. eprint 2006/291 (2006)
19. Cox, D.A.: Primes of the form x2 + ny2: Fermat, class field theory, and complex multiplication. Wiley (1997)

20. De Feo, L.: Mathematics of isogeny based cryptography. Notes from a summer school on Mathematics for Post-quantum cryptography (2017), https://arxiv.org/abs/1711.04062
21. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3), 209–247 (2014)
22. De Feo, L., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. In: Peyrin, T., Galbraith, S.D. (eds.) ASIACRYPT 2018. Lecture Notes in Computer Science, vol. 11274, pp. 365–394. Springer (2018)
23. Decru, T., Panny, L., Vercauteren, F.: Faster SeaSign signatures through improved rejection sampling. to appear at PQCrypto 2019 (2019)
24. Delfs, C., Galbraith, S.D.: Computing isogenies between supersingular elliptic curves over F_p. Des. Codes Cryptography **78**(2), 425–440 (2016)
25. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. Lecture Notes in Computer Science, vol. 8042, pp. 40–56. Springer (2013)
26. Fukase, M., Kashiwabara, K.: An accelerated algorithm for solving SVP based on statistical analysis. Journal of Information Processing **23**(1), 67–80 (2015)
27. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press (2012)
28. Galbraith, S.D., Hess, F., Smart, N.P.: Extending the GHS weil descent attack. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 29–44. Springer (2002)
29. Galbraith, S.D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. Lecture Notes in Computer Science, vol. 10624, pp. 3–33. Springer (2017)
30. Hafner, J.L., McCurley, K.S.: A rigorous subexponential algorithm for computation of class groups. Journal of the American mathematical society **2**(4), 837–850 (1989)
31. Huelsing, A., Butin, D., Gazdag, S.L., Rijneveld, J., Mohaisen, A.: XMSS: eXtended Merkle Signature Scheme. RFC 8391 (May 2018)
32. Hülsing, A., Rijneveld, J., Song, F.: Mitigating multi-target attacks in hash-based signatures. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) Public-Key Cryptography – PKC 2016. pp. 387–416. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
33. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B. (ed.) PQCrypto 2011. Lecture Notes in Computer Science, vol. 7071, pp. 19–34. Springer (2011)
34. Jao, D., LeGrow, J., Leonardi, C., Ruiz-Lopez, L.: A subexponential-time, polynomial quantum space algorithm for inverting the CM group action. To appear in proceedings of MathCrypt (2019)
35. Jao, D., Soukharev, V.: A subexponential algorithm for evaluating large degree isogenies. In: ANTS. pp. 219–233 (2010)
36. Jaques, S., Schanck, J.M.: Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. Cryptology ePrint Archive, Report 2019/103 (2019), https://eprint.iacr.org/2019/103
37. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. Lecture Notes in Computer Science, vol. 10822, pp. 552–586. Springer (2018)
38. Kitaev, A.Y.: Quantum measurements and the abelian stabilizer problem. arXiv preprint quant-ph/9511026 (1995), https://arxiv.org/abs/quant-ph/9511026
39. Kuperberg, G.: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM Journal of Computing **35**(1), 170–188 (2005)
40. Lindner, R., Peikert, C.: Better key sizes (and attacks) for lwe-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. Lecture Notes in Computer Science, vol. 6558, pp. 319–339. Springer (2011)
41. Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer (2009)
42. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 738–755. Springer (2012)
43. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) Advances in Cryptology — CRYPTO' 89 Proceedings. pp. 218–238. Springer New York, New York, NY (1990)
44. Meyer, M., Reith, S.: A faster way to the CSIDH. In: Chakraborty, D., Iwata, T. (eds.) INDOCRYPT 2018. Lecture Notes in Computer Science, vol. 11356, pp. 137–152. Springer (2018)
45. National Institute of Standards and Technology: Announcing request for nominations for public-key post-quantum cryptographic algorithms (2016), https://www.federalregister.gov/d/2016-30615
46. Neven, G., Smart, N.P., Warinschi, B.: Hash function requirements for Schnorr signatures. J. Mathematical Cryptology **3**(1), 69–87 (2009)
47. Regev, O.: A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv:quant-ph/0406151 (Jun 2004), http://arxiv.org/abs/quant-ph/0406151

48. Renes, J.: Computing isogenies between montgomery curves using the action of (0, 0). In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 229–247. Springer International Publishing, Cham (2018)
49. Shanks, D.: On Gauss and composition. In: Number Theory and Applications. pp. 163–204. NATO – Advanced Study Institute, Kluwer Academic Press (1989)
50. Silverman, J.H.: The arithmetic of elliptic curves, GTM, vol. 106. Springer (1986)
51. Stolbunov, A.: Cryptographic schemes based on isogenies. Doctoral thesis, NTNU (2012)
52. Sutherland, A.: Elliptic curves. Lecture notes from a course (18.783) at MIT (2017), `http://math.mit.edu/classes/18.783/2017/lectures`
53. Vélu, J.: Isogénies entre courbes elliptiques. Comptes Rendus de l'Académie des Sciences de Paris **273**, 238–241 (1971)
54. Washington, L.C.: Elliptic curves: Number theory and cryptography, 2nd ed. CRC Press (2008)
55. Yoo, Y., Azarderakhsh, R., Jalali, A., Jao, D., Soukharev, V.: A post-quantum digital signature scheme based on supersingular isogenies. In: Kiayias, A. (ed.) Financial Cryptography and Data Security FC 2017. Lecture Notes in Computer Science, vol. 10322, pp. 163–181. Springer (2017)