Quantum Security Analysis of CSIDH and Ordinary Isogeny-based Schemes

Xavier Bonnetain[1,2] and André Schrottenloher[2]

[1] Sorbonne Université, Collège Doctoral, F-75005 Paris, France
[2] Inria, France

Abstract. CSIDH is a recent proposal by Castryck, Lange, Martindale, Panny and Renes for post-quantum non-interactive key-exchange, to be presented at ASIACRYPT 2018. It is similar in design to a scheme by Couveignes, Rostovtsev and Stolbunov, but it replaces ordinary elliptic curves by supersingular elliptic curves, in order to make significant gains in time and key lengths.
Isogeny-based key-exchange on ordinary elliptic curves can be targeted by a quantum subexponential hidden shift algorithm found by Childs, Jao and Soukharev. Although CSIDH uses supersingular curves, it is analog to the case of ordinary curves, hence this algorithm applies.
In the proposal, the authors suggest a choice of parameters that should ensure security against this.
In this paper, we reassess these security parameters. Our result relies on two steps: first, we propose a new quantum algorithm for the hidden shift problem and analyze precisely its complexity. This reduces the number of group actions to compute w.r.t the authors' estimation; second, we show how to compute efficiently this group action.
For example, we show that only 2^{35} quantum equivalents of a key-exchange are sufficient to break the 128-bit classical, 64-bit quantum security parameters proposed, instead of 2^{62}.
Finally, we extend our analysis to ordinary isogeny computations, and show that an instance proposed by De Feo, Kieffer and Smith (also accepted at ASIACRYPT 2018) and expected to offer 56 bits of quantum security can be attacked in 2^{38} quantum evaluations of a key exchange.

Keywords: Post-quantum cryptography, isogeny-based cryptography, quantum cryptanalysis, hidden shift problem, lattices

1 Introduction

Post-quantum Security. Problems such as factoring and solving discrete logarithms, believed to be classically intractable, underlie the security of most asymmetric cryptographic primitives in use today in digital signatures, key exchange, and so on. Since Shor [31] obtained a quantum polynomial-time algorithm for both, the cryptographic community has been actively working on replacements, culminating with the ongoing NIST call for post-quantum primitives [28].

Isogeny-based Protocols. While the elliptic curve Discrete Logarithm Problem is one of the targets of Shor's algorithm, there have been proposals at post-quantum asymmetric primitives using elliptic curve *isogenies*. Informally speaking, isogenies are morphisms between elliptic curves and the security of these schemes is related to the difficulty of finding an isogeny between two given curves.

The first proposals used *ordinary* elliptic curves, but a quantum algorithm for constructing isogenies between them was found in [11]. It runs in *subexponential* time, albeit not polynomial like Shor's algorithm, while all known classical attacks require exponential time. Indeed, the action of isogenies on ordinary curves is a commutative group action: it is possible to break it and recover a secret isogeny using a subexponential number of evaluations of this group action, via Kuperberg's algorithm [24].

As a consequence, there has been a move towards isogeny-based cryptography using supersingular elliptic curves [17]. The NIST candidate SIKE uses this principle. In general, in that case, the previous algorithm does not apply, because supersingular isogenies do not yield such a "global" structure as the classical ones. However, much work is still required to understand precisely the hardness of supersingular isogeny problems against quantum computers (see [20] for a survey).

CSIDH. CSIDH is a new asymmetric primitive proposed in [7] as a replacement for elliptic curve cryptography. It uses supersingular elliptic curves, with the additional constraint that they have to be defined over $\mathbb{F}_p$. This case turns out to be analog to ordinary curves: in particular, as there is a commutative group action on them, the subexponential quantum attack of [11] still applies. This is taken into account by the authors of [7]. Their quantum security claims are based on a query complexity of:

$$\exp\left((\sqrt{2}+o(1))\sqrt{\log N \log\log N}\right)$$

where $N = O(\sqrt{p})$ is the size of the class group and p is the prime number used for the base field. This complexity represents the number of evaluations of the group action required to break the scheme.

This complexity is taken from [11] and the study of an algorithm by Regev, designed to run in polynomial quantum space.

Even if there is, as with the schemes based on ordinary curves, an attack of subexponential complexity, the structure of CSIDH allows to lower considerably the computational cost of the scheme, obtaining an improved balance of efficiency vs. quantum security. The query complexity given above is used in [7] to derive quantum securities, without taking into account the cost of a query (roughly speaking, the evaluation of a group action). The authors provide three sets of parameters, to meet three security levels of the NIST call: levels 1, 3 and 5, respectively equivalent to performing a key-recovery on AES-128, AES-192 and AES-256. The corresponding base field cardinality p has size 512, 1024 and 1792 bits, respectively. They leave a precise security analysis as an open problem.

Further Work. Since its publication, CSIDH has been the subject of active research, e.g on improved implementations [27] or on how to use its group action for signatures [16]. We note that in this work, we are targeting the most essential building block of CSIDH, its commutative group action; hence our security analysis extends to other contexts where this scheme is applied. Moreover, the quantum circuits necessary to evaluate the CSIDH oracle in the attack have been studied in [2]: this allows to derive a precise gate count for the procedures detailed in this paper, while we primarily focus on query complexity.

Contents.

Quantumly Attacking CSIDH. In this paper, we show that the parameters proposed initially in [7] offer less quantum security than expected, respectively less than 35, 48 and 63 bits of security instead of 62, 94 and 129. [3]

This is a consequence of two crucial points: first, we rely on a precise (i.e without asymptotics) complexity estimate of Kuperberg's algorithm, which uses less queries than estimated in [7]. It is worth to notice that [7] considers a polynomial-space algorithm more relevant; we go for the best time complexity instead, using subexponential quantum memory, which we quantify as well.

Second, we show that evaluating efficiently the commutative group action adds only little overhead w.r.t a legitimate key-exchange, using lattice reduction methods that date back to Couveignes [13]. These methods are not new: they were used in [3] in order to compute efficiently large-degree isogenies and were already mentioned in [7] (Section 7.2, "subexponential vs. practical") as a potential improvement.

Quantum Security. We consider two metrics for quantum security. The CSIDH parameter sizes required to meet the expected security levels differ between these two metrics, so we present both.

First, we use as metric the gap between a legitimate use (here, a key exchange or group action evaluation) and the attack: we give complexities in multiples of the cost of a key-exchange using CSIDH. Second, in Section 7, we elaborate on the cost when compared to the levels of security proposed in the NIST post-quantum call. These metrics require to count the number of quantum gates of the attack algorithm and compare against Grover exhaustive search of secret keys for AES instances. Since the first version of our work, this has been done in full detail in [2]. We now use these results to benchmark the original CSIDH parameters against the NIST levels.

[3] Before making this work public, we contacted the authors of [7]. They agreed with our analysis. Since then, their work has been accepted to ASIACRYPT 2018. This paper is based on the eprint version of their article [8], last revised 11 may 2018. Unfortunately, at the time of submission, we didn't have access to the final version, so we couldn't take into account possible changes. As we do not know if the parameters proposed were updated accordingly to our estimations, all assessments of the CSIDH parameters refer only to those of [8].

Concrete Estimates for Ordinary Isogeny Schemes. Finally, we extend our approach to ordinary isogenies, and are able to propose better-than-expected costs estimates for a scheme proposed by De Feo, Kieffer and Smith in [18].

Paper Outline.

Section 2 below presents the CSIDH group action, some of its mathematical background and recalls the results from [7] (to which we refer for more details). Section 3 presents the outline of the quantum subexponential attack on ordinary-looking isogeny-based schemes. The two parts of the attack are then developed in the following sections. Section 4 proposes a a new algorithm for the Hidden Shift problem for cyclic groups and estimates its cost. Section 5 presents some lattice reduction methods. In Section 6, we extend our study to the CRS scheme (based on ordinary curves) and the scheme of [18] and show how to adapt the lattice steps. Finally, in Section 7, we summarize our complexity analysis and elaborate on the NIST metric.

2 Preliminaries

In this section, we briefly describe the design principles underlying the CSIDH group action and its rationale, and recall the security claims of [7].

2.1 Context

Let p be a prime number. In general, supersingular elliptic curves over $\overline{\mathbb{F}_p}$ are defined over a quadratic extension $\mathbb{F}_{p^2}$. However, the case of supersingular elliptic curves *defined over* $\mathbb{F}_p$ is special: there is a correspondence with ordinary elliptic curves of $\mathbb{F}_p$-endomorphism ring $\mathbb{Z}[\sqrt{-p}]$ or $\mathcal{O}_K$, with K a quadratic imaginary field. This fact is proven in [14], where we find it first exploited to compute isogenies between supersingular elliptic curves.

In [5], this correspondence gives rise to a quantum algorithm for computing such isogenies. When $\mathcal{O}$ is an order in an imaginary quadratic field, each supersingular elliptic curve defined over $\mathbb{F}_p$, with $\mathcal{O}$ as $\mathbb{F}_p$-rational endomorphism ring, corresponds to an ideal class in $\mathcal{C}\ell(\mathcal{O})$. Moreover, a rational ℓ-isogeny from such a curve corresponds to an ideal of norm ℓ in $\mathcal{C}\ell(\mathcal{O})$. The (commutative) class group $\mathcal{C}\ell(\mathcal{O})$ acts on the set of supersingular elliptic curves with $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O}$.

This group action exists already in the case of ordinary curves and it gives rise to a subexponential quantum attack, based on Kuperberg's hidden shift algorithm [11], which recovers a "hidden" isogeny between two given curves. The same attack applies in this particular supersingular case, however, the balance between cost and security is different, as the CSIDH scheme allows for less intensive computations and enables easy key validation.

All use cases of the CSIDH scheme can be pinned down to the definition of a *one-way group action.*[4] A group G acts on a set X. Operations in G are easy to compute and the action $g * x$ for $g \in G, x \in X$ is easy to compute, while recovering g given x and $x' = g * x$ is hard. In the case of CSIDH, X is a set of elliptic curves over $\mathbb{F}_p$, and the group G is $\mathcal{C}\ell(\mathcal{O})$ for $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. Taking $g * x$ for an element in $\mathcal{C}\ell(\mathcal{O})$ (i.e an isogeny) and a curve corresponds to computing the image curve of x by this isogeny.

Furthermore, the scheme is defined so that:

- The curves (public keys or parameters) are efficiently represented;
- The isogenies / elements of $\mathcal{C}\ell(\mathcal{O})$ (private keys or parameters) are efficiently represented;
- The action of an element in $\mathcal{C}\ell(\mathcal{O})$ can be efficiently computed.

We present below the design strategy which enables CSIDH to reach these requirements.

2.2 Design of the CSIDH Group Action

First, we consider a very specific set of elliptic curves.

Let E_0 be the curve $y^2 = x^3 + x$ with endomorphism ring $\mathcal{O} = \mathbb{Z}[\pi]$ (π is the Frobenius endomorphism, with $\pi^2 = -p$ since we consider supersingular curves). The following proposition is proven in [7]:

Proposition 1 (Prop. 8 in [7]). *Let $p \equiv 3 \mod 8$ and let E be a supersingular elliptic curve over $\mathbb{F}_p$. Then $End_p(E) = \mathbb{Z}[\pi]$ iff there exists $A \in \mathbb{F}_p$ such that E is $\mathbb{F}_p$-isomorphic to the curve $E_A : y^2 = x^3 + Ax^2 + x$. Moreover, if such an A exists, then it is unique.*

It shows that by considering the action of $\mathcal{C}\ell(\mathcal{O})$ on elliptic curves which are $\mathbb{F}_p$-supersingular, with endomorphism ring $\mathbb{Z}[\pi]$, one obtains Montgomery curves, which can be represented by a single element in $\mathbb{F}_p$.

Short Representations of Classes. The prime p is chosen with a very specific form: $p = 4 \cdot \ell_1 \cdots \ell_u - 1$ is selected, where $\ell_1, \ldots, \ell_u$ are small primes. The number u should be chosen the highest possible in order to speed up the key exchange.

Due to the special form of the prime p chosen, the elements of $\mathcal{C}\ell(\mathcal{O})$ (which correspond to isogenies) can be represented in a way that allows for *fast computation* of the corresponding isogenies. Indeed, since each of the ℓ_i divides $-p - 1 = \pi^2 - 1$, the ideal $\ell_i\mathcal{O}$ splits and $\mathfrak{l}_i = (\ell_i, \pi - 1)$ is an ideal in $\mathcal{O}$.

We now consider in $\mathcal{C}\ell(\mathcal{O})$ products of the form:

$$\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$$

[4] This is also the definition of a *hard homogeneous space* by Couveignes [13].

where $e_i \in \{-m \dots m\}$ for some small m, and $[\mathfrak{l}_i]$ is the class of $\mathfrak{l}_i$. It is easy to see that, even with a small m, this method will span the whole group $\mathcal{C}\ell(\mathcal{O})$ (or almost all of it) using these products. For example, for the quantum 64-bit security example of [7], $m = 5$. More generally, we take $2m + 1 \simeq p^{1/(2u)}$. The only assumption is that products of the form above fall randomly in $\mathcal{C}\ell(\mathcal{O})$, which has $O(\sqrt{p})$ elements. With the fact that u should be the greatest possible, we derive optimal parameters for the three security levels from [7] in Table 1.

Table 1: Approximate parameters for the three security levels of [7].

Level	Expected quantum security	$\log_2 p$	u	m
NIST 1	64	512	74	5
NIST 3	96	1024	132*	7*
NIST 5	128	1792	209*	10*

*In [7], only parameters for the first instance are given. We consider the biggest possible u, as it makes the scheme more efficient, and our attack more costly.

Once we know the decomposition of an ideal as a product $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$ for $-m \leq e_i \leq m$, we may simply represent it as the sequence $\bar{e} = (e_1 \dots e_u)$.

Computing with Ideal Classes. Computing operations in the class group is easy and only costs multiplications and inversions modulo p. There exists a canonical way of representing an ideal class (using the theory of reduction of quadratic forms). We do not go into details here.

Computing the Class Group Action. Given an element of $\mathcal{C}\ell(\mathcal{O})$ of the form $[\mathfrak{b}] = \prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$, we use this representation to compute $E' = [\mathfrak{b}] \cdot E$, which is simply the image curve of E by the isogeny represented by $[\mathfrak{b}]$. The authors of [7] provide (Section 8) an efficient implementation of the class group action, which does not need to compute modular polynomials (usually the most costly operation).

2.3 Claimed Security of CSIDH

We review the security claims of CSIDH in [7]. The main problem considered is, given a Montgomery curve E_A, recovering the isogeny $[\mathfrak{b}] \in \mathcal{C}\ell(\mathcal{O})$ such that $E_A = [\mathfrak{b}] \cdot E_0$. Moreover, the ideal $\mathfrak{b}$ that represents it should be sufficiently "small", so that the action of $[\mathfrak{b}]$ on a curve can be evaluated. Otherwise, this secret key would be of no use to the adversary.

The authors study different ways of recovering $[\mathfrak{b}]$. The complexity of these methods depends on the size of the class group $\mathcal{C}\ell(\mathcal{O})$, which is $O(\sqrt{p})$.

- Classically, the best method seems the exhaustive key search of $[\mathfrak{b}]$ using a meet-in-the-middle approach: it costs $O(p^{1/4})$.

- Quantumly, the best attack comes from [11], using Kuperberg's algorithm to recover a *hidden shift* in a commutative group action. The group is $\mathcal{C}\ell(\mathcal{O})$ and the cost claimed is:

$$\exp\left((\sqrt{2} + o(1)) \sqrt{\log N \log\log N} \right) ,$$

  where $N = \#\mathcal{C}\ell(\mathcal{O})$, which actually counts how many times one has to evaluate the action of $\mathcal{C}\ell(\mathcal{O})$ on the curves E_A and E_0, in superposition over all elements of this group. The authors from [11] give a technique to do this in time subexponential in p, but this induces a significant overhead.

Complexity Analysis. We evaluate the complexity of our quantum procedures in terms of corresponding classical class group actions. More precisely, we consider that computing the action of $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$ on a curve costs $O(\sum_{i=1}^{u} |e_i|) = O(||(e_1 \dots e_u)||_1)$, as this is the number of small isogenies computed. The constant overhead depends on the computation for *one* small isogeny.

Since the classical protocol computes the action of $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$ where $|e_i| \leq m$ for all i and a constant m, its cost is $O(um)$. It is the unit of the classical security estimates given in [7]. We shall adopt the exact same benchmark for quantum securities. In particular, suppose that we compute a group action of an element of L_1 norm N; we will count this cost as $N/(um)$.

Remark 1. Since we will evaluate the class group action in quantum superposition, it is possible that the efficient method of [7], Section 8,would perform only at its worst time complexity. This it is not a significant issue, especially as a concrete implementation of the key exchange would require to evaluate the isogenies in constant time to avoid timing attacks.

Remark 2. A more refined analysis could take into account that the small ideal classes $[\mathfrak{l}_i]$ represent isogenies of increasing degrees (albeit in a small range). We put this consideration aside for the sake of simplicity.

3 Attack Outline

We suppose given access to an offline quantum computer and wish, given E_A, to find $[\mathfrak{b}]$ such that $E_A = [\mathfrak{b}] \cdot E_0$. We do not exactly retrieve the secret key $\bar{e}$ which was selected at the beginning, but we find an alternative vector $\bar{e}'$ whose norm L_1 is bounded by $||\bar{e}||_1$ multiplied by a "small" factor. This is enough for practical purposes: using the equivalent secret key, i.e computing the corresponding isogeny, will cost more than for the legitimate user, but only by this same factor.

Ideas. The best quantum attack on CSIDH, which we estimate below, makes use of already available ideas, being found in [7], [13] and [3]:

- Kuperberg's algorithm was already considered in [7, Section 7.2], but the quantum complexity analysis was only partial. We propose a new time-efficient variant of this algorithm.

- Using lattice reduction to compute efficiently the group action was mentioned in [7] and [32] as a possible method, and already done in [13] in a very similar setting, albeit a classical one. In [3], lattice reduction techniques are used to obtain short representations of large-degree isogenies as elements of $\mathcal{C}\ell(\mathcal{O})$. This direction is further followed in [4] where, independently from our work, the authors study how to attack the CSIDH scheme using this efficient isogeny evaluation oracle. As they are more interested in the space complexity and asymptotic time complexity of the algorithms, they use a method different from us, without relying at all on the ideal classes given by the scheme.

The attack runs in two phases. The first one is a precomputation which depends only on the public parameters p and $\ell_1, \dots \ell_u$. It does not depend on the particular public key targeted. Consequently, there could be a trade-off between the complexity of this first phase and the second one, where the adversary performs a secret-key recovery. For the proposed parameters, the first phase only requires a small amount of quantum and classical computations, and the cost of our attack is the cost of the second phase.

Phase 1: Computing a Short Basis of Multi-periods. Given $\ell_1, \dots \ell_u$ and the ideal classes $[\mathfrak{l}_1], \dots, [\mathfrak{l}_u]$, we compute an approximate short basis of the lattice of all relations $\bar{f} = f_1, \dots f_u$ such that $[\mathfrak{l}_1]^{f_1} \dots [\mathfrak{l}_u]^{f_u} = 1$. For comprehensiveness, we call such relations "multi-periods" by analogy with period-finding. This phase uses quantum order-finding (of polynomial complexity) and a classical lattice reduction. For the given parameters, this reduction seems to be doable on a standard computer.

Phase 2: Solving the Hidden Shift Problem. Using Kuperberg's algorithm, we solve a hidden shift problem instance: finding the isogeny $[\mathfrak{b}]$ such that for each $[x]$ in the class group $\mathcal{C}\ell(\mathcal{O})$:

$$[x] \cdot [\mathfrak{b}] \cdot E_0 = [x] \cdot E_A \ .$$

This requires to compute $[x] \cdot E$ for $[x]$ in $\mathcal{C}\ell(\mathcal{O})$, in superposition over $[x]$, for some curve E.

To do this, we first rewrite $[x]$ on the set $[\mathfrak{l}_1], \dots, [\mathfrak{l}_u]$: $[x] = [\mathfrak{l}_1]^{x_1} \dots [\mathfrak{l}_u]^{x_u}$ using quantum order-finding and *ad hoc* computations. Then, we use the short basis of phase 1 to solve the approximate closest vector problem, enabling us to represent $[x]$ by a "smaller" product $[x] = [\mathfrak{l}_1]^{y_1} \dots [\mathfrak{l}_u]^{y_u}$, with shorter coefficients. The complexity of this computation is controlled by the quality of our approximation, which we relate to the quality of the basis obtained via lattice reduction.

4 Hidden Shift

The Hidden Shift Problem. The hidden shift problem is defined as follows.

Problem 1 (Hidden shift problem). Let $(\mathbb{G}, +)$ be a group, $f, g : \mathbb{G} \to \mathbb{G}$ two permutations such that there exists $s \in \mathbb{G}$ such that, for all x, $f(x) = g(x + s)$. Find s.

Classically, this problem essentially reduces to a collision search, but in the case of abelian groups, quantum subexponential algorithms exists. In [7], 7.2, the authors consider a *query* complexity to solve this problem of:

$$L_N(1/2, \sqrt{2} + o(1)) = \exp\left((\sqrt{2} + o(1))\sqrt{\log N \log\log N}\right)$$

where $N = \#cl(\mathcal{O})$, as there is a hidden shift structure in the class group action.

Quantum Algorithms. There are multiple results on solving the hidden shift problem in commutative groups. The first result is an efficient algorithm in query, by Ettinger and Høyer [15], which needs $O(\log(N))$ queries and $O(N)$ classical computations to solve the hidden shift in $\mathbb{Z}/N\mathbb{Z}$. The first time-efficient algorithms were proposed by Kuperberg in [24], where Algorithm 3 is shown to have a complexity in quantum queries and memory of $\widetilde{O}\left(2^{\sqrt{2\log_2(3)\log_2(N)}}\right)$ for the group $\mathbb{Z}/N\mathbb{Z}$ for smooth N, and Algorithm 2 is in $O\left(2^{3\sqrt{\log_2(N)}}\right)$, for any N. This has been followed by a memory-efficient variant by Regev, with a query complexity in $L_N(1/2, \sqrt{2})$ and a polynomial memory complexity, in [29], which has been generalized by Kuperberg in [25], with an algorithm heuristically in $\widetilde{O}(2^{\sqrt{2\log_2(N)}})$ quantum queries and classical memory with quantum access, and a polynomial quantum memory. Regev's variant has been generalized to arbitrary commutative groups in the appendix of [11], with the same complexity.

Concrete Estimates. The $\widetilde{O}$ are not practical for concrete estimates. However, in [6], the authors showed that the polynomial of a variant of Kuperberg's original algorithm is a constant around 1 if N is a power of 2, and that the problem is easier if the group is not one big cyclic group. As we will not have an N which is a power of 2, we propose in what follows a generalization of [6, Algorithm 2] that works for any N, at essentially the same cost.

Memory Cost. The algorithm we consider has a subexponential memory cost. More precisely, it needs exactly one qubit per query, plus the fixed overhead of the oracle, which can be neglected. This is a fairly important memory usage. To take this into account, we could take other metrics, like the time-memory product (which is very constraining, as a parallelized Grover's algorithm has an increased time-memory product cost), or a time-square-memory product (which corresponds to a parallelized Grover). We believe that considering the most time-efficient variant is relevant, since there may be trade-off algorithms between the time-efficient and memory-efficient variants [25].

Moreover, Kuperberg's algorithm does not need to have a highly entangled memory for a long time, and only performs operations on 2 qubits at a time. If some labeled qubits are detected as noisy, one may choose to avoid using it in the remaining of the computation.

Application to Commutative Group Action. The hidden shift problem can be used to retrieve an isogeny given its origin curve E_0 and its image curve E_1. The isogeny corresponds to an element s of the class group that verifies $[s]E_0 = E_1$. Moreover, as we have a group action, we have, for all element of the class group, $[x][s]E_0 = [x]E_1$. This is an instance of the hidden shift problem, in the class group. In order to apply Kuperberg's algorithm, we first need to have a representation of the class group. We can use the quantum polynomial-time algorithm of [10], as done in [11]. We are only interested in the subgroup spanned by $([\mathfrak{l}_1], \dots, [\mathfrak{l}_u])$ (which should be close to the total group): we can get it using the same method. We then obtain a generating set $([\mathfrak{g}_1], \dots, [\mathfrak{g}_g])$.

The problem is now to efficiently compute an isogeny given its representation $[\mathfrak{g}_1]^{e_1} \dots [\mathfrak{g}_g]^{e_g}$.

Efficiently Applying Kuperberg's Algorithm. The idea is to use the same representation as for the key exchange, that is, given $([\mathfrak{l}_1], \dots, [\mathfrak{l}_u])$, express any element $[\mathfrak{g}_1]^{e_1} \dots [\mathfrak{g}_k]^{e_k}$ as a product of the $[\mathfrak{l}_i]$, and reduce the L_1 norm of the element in this representation. As one query requires to compute all the possible isogenies in quantum superposition, the cost per query depends on this L_1 norm.

The hidden shift problem will yield an element of the class group such that $[s]E_0 = E_1$, that is, the wanted isogeny. We can then use the same method to obtain a representation of this isogeny in the basis $([\mathfrak{l}_1], \dots, [\mathfrak{l}_u])$. This may not be the smallest possible isogeny, but it will be small enough to be usable. This is Algorithm 1. Its cost in quantum query and memory is the one of the hidden shift algorithm, which is summarized in Table 3.

Algorithm 1 Key Recovery

```
   Input: The elements $([\mathfrak{l}_1], \dots, [\mathfrak{l}_u])$, two curves $E_0$ and $E_A$ defined over $\mathbb{F}_p$, a generating set of $\mathcal{C}\ell(\mathcal{O})$: $([\mathfrak{g}_1], \dots, [\mathfrak{g}_k])$
   Output: A vector $(e_1, \dots, e_u)$ such that $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i} \cdot E_0 = E_A$
1: Define $f : [\mathfrak{g}] \in \mathcal{C}\ell(\mathcal{O}) \mapsto [\mathfrak{g}] \cdot E_0$ and $g : [\mathfrak{g}] \in \mathcal{C}\ell(\mathcal{O}) \mapsto [\mathfrak{g}] \cdot E_A$ , to be computed with Algorithm 5.
2: Apply Algorithm 2 on $f$ and $g$, get $[\mathfrak{s}]$.
3: Using Algorithm 5, decompose $[\mathfrak{s}]$ as $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$ with small $e_i$.
4: return $(e_1, \dots, e_u)$
```

We need to add the cost per query to obtain the total cost. Each query needs to compute $[x] \cdot E$ for all $[x]$ in the group generated by $([\mathfrak{l}_1], \dots, [\mathfrak{l}_u])$ and some curve E. This cost will be studied in the next sections.

4.1 Hidden Shift Algorithm for Cyclic Groups

In this section, we present a generic hidden shift algorithm for $\mathbb{Z}/N\mathbb{Z}$, which allows us to have the concrete estimates we need.

We suppose an access to the quantum oracle

$$\begin{array}{c} |x\rangle |0\rangle |y\rangle \mapsto |x\rangle |0\rangle |y \oplus f(x)\rangle \\ |x\rangle |1\rangle |y\rangle \mapsto |x\rangle |1\rangle |y \oplus g(x)\rangle \end{array}.$$

We begin by constructing the uniform superposition on $N \times \{0,1\}$, that is,

$$\frac{1}{\sqrt{2N}} \sum_{x=0}^{N-1} |x\rangle \left(|0\rangle + |1\rangle\right) |0\rangle .$$

Then, we apply the quantum oracle, and get

$$\frac{1}{\sqrt{2N}} \sum_{x=0}^{N-1} |x\rangle \left(|0\rangle |f(x)\rangle + |1\rangle |g(x)\rangle\right) .$$

We then measure the final register, to obtain an $f(x_0) = g(x_0 + s)$ and the qubit

$$\frac{1}{\sqrt{2}} \left(|x_0\rangle |0\rangle + |x_0 + s\rangle |1\rangle\right) .$$

Finally, we apply a quantum Fourier Transform on the first register and measure it, we obtain a label ℓ and the state

$$|\psi_\ell\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \chi\left(s\frac{\ell}{N}\right)|1\rangle\right), \chi\left(x\right) = \exp\left(2i\pi x\right) .$$

This phase depends of s and $\frac{\ell}{N}$, and by applying a CNOT and measuring the second qubit, we can combine two qubits ℓ_1 and ℓ_2 into $\ell_1 + \ell_2$ or $\ell_1 - \ell_2$. This combination is destructive, and we only now which output it is after it.

In order to retrieve s, we want to produce the qubits with label 2^i and apply a Quantum Fourier Transform (QFT). Indeed, we have

$$QFT \bigotimes_{i=0}^{n-1} |\psi_{2^i}\rangle = \frac{1}{2^{n/2}} QFT \sum_{k=0}^{2^n-1} \chi\left(\frac{ks}{N}\right)|k\rangle = \frac{1}{2^n} \sum_{t=0}^{2^n-1} \left(\sum_{k=0}^{2^n-1} \chi\left(k\left(\frac{s}{N} + \frac{t}{2^n}\right)\right)\right)|t\rangle .$$

The amplitude associated with t is $\frac{1}{2^n}\left|\frac{1-\chi\left(2^n\left(\frac{s}{N}+\frac{t}{2^n}\right)\right)}{1-\chi\left(\frac{s}{N}+\frac{t}{2^n}\right)}\right|$. If we note $\theta = \frac{s}{N} + \frac{t}{2^n}$, this amplitude is $\frac{1}{2^n}\left|\frac{\sin(2^n\pi\theta)}{\sin(\pi\theta)}\right|$. For $\theta \in [0; \frac{1}{2^{n+1}}]$, this value is decreasing, from 1 to $\frac{1}{2^n \sin(\frac{\pi}{2^{n+1}}})\simeq \frac{2}{\pi}$.

Hence, when measuring, we obtain a t such that $\left|\frac{s}{N} + \frac{t}{2^n}\right| \leq \frac{1}{2^{n+1}}$ with probability greater than $\frac{4}{\pi^2}$. Such a t always exists, and uniquely defines s if $n > \log_2(N)$.

In order to make these qubits, the idea is then to apply Kuperberg's algorithm, but to not take into account the modulo, that is, obtain a combination such that $\sum_k \pm\ell_k = 2^i$. By applying the algorithm, we perform some collisions on the lowest significant bits, but we also increase the maximum size. However, the size can increase of at most one bit per combination, while the lowest significant 1 position increases on average in $\sqrt{n}$. Hence, the algorithm will eventually produce the correct value.

We note $\mathrm{val}_2(x) = \max_i 2^i | x$ the 2-valuation of x. In Algorithm 2, each label is associated to its corresponding qubit, and the operation $\pm$ corresponds to the combination.

Algorithm 2 Hidden shift algorithm for $\mathbb{Z}/N\mathbb{Z}$

```
    **Input:** $N$, a number of queries $Q$, a quantum oracle access to $f$ and $g$ such that $f(x) = g(x+s), x \in \mathbb{Z}/N\mathbb{Z}$
    **Output:** $s$
 1: Generate $Q$ random labels in $[0; N)$ using the quantum oracles
 2: Separate them in pools $P_i$ of elements $e$ such that $\mathrm{val}_2(x) = i$
 3: $i \leftarrow 0$
 4: $R = \emptyset$
 5: $n \leftarrow \lfloor \log_2(N) \rfloor$.
 6: while some elements remain do
 7:     if $i \le n$ then
 8:         Pop a few elements $e$ from $P_i$, put $(e, i)$ in $R$.
 9:     end if
10:     for $(e, j) \in R$ do
11:         if $\mathrm{val}_2(e - 2^j) = i$ then
12:             Pop $a$ of $P_i$ which maximizes $\mathrm{val}_2(a + e - 2^j)$ or $\mathrm{val}_2(e - 2^j - a)$
13:             $e = e \pm a$
14:         end if
15:     end for
16:     if $\{(2^i, i) | 0 \le i \le n\} \subset R$ then
17:         Apply a QFT on the qubits, measure a $t$
18:         $s \leftarrow \left\lceil \frac{-Nt}{2^{n+1}} \right\rfloor \mod N$
19:         return $s$
20:     end if
21:     while $|P_i| \ge 2$ do
22:         Pop two elements $(a, b)$ of $P_i$ which maximizes $\mathrm{val}_2(a + b)$ or $\mathrm{val}_2(a - b)$
23:         $c = a \pm b$
24:         Insert $c$ in the corresponding $P_j$
25:     end while
26:     $i \leftarrow i + 1$
27: end while
28: return Failure
```

Intuitively, the behaviour of this algorithm will be close to the one of [6], as we only have a slightly higher amplitude in the values, and a few more elements to produce.

In practice, we can simulate this algorithm, to estimate its complexity. Empirically, we only need to put 3 elements at each step in R in order to have a good success probability.

$\log_2(N)$	$\log_2(Cost)$	$1.8\sqrt{\log_2(N)} + 2.3$
20	10.1	10.3
32	12.4	12.5
50	15.1	15.0
64	16.7	16.7
80	18.4	18.4
100	20.3	20.3

Table 2: Simulation results for Algorithm 2, for 90% success

We can estimate the total complexity to be around $12 \times 2^{1.8\sqrt{n}}$, heuristically.

Table 3: Cost estimates for the hidden shift algorithm.

$\log_2(p)$	n	Hidden shift cost ($\log_2$)	Memory cost ($\log_2$)
512	256	32.5	31
1024	512	44.5	43
1792	896	57.5	56

4.2 Generalization to products of cyclic groups

In [6], an algorithm for product of cyclic groups was proposed, with a cost that can be drastically below the cyclic case, if there were many small groups. We study here the situation of product groups without any constraint on the size, with the group $\mathbb{Z}/N_1\mathbb{Z} \times \cdots \times \mathbb{Z}/N_m\mathbb{Z}$. The same approach than in [6] seems risky here: the proposed algorithm performed some combinations such that one bit was gained on each component. If we try to do the same here, we could obtain bigger numbers in each component, while only gaining a one bit per component. Hence, we rather have the exact same approach as Algorithm 2, but instead of only focusing on one component, we address them all, sequentially.

We have some elements $|\psi_{\ell_1,\ldots,\ell_m}\rangle$, and the addition/subtractions are replaced by the termwise ones. The labels become some vectors $(\ell_1, \ldots, \ell_m)$. The objective, at the end, is to obtain the labels $(0, \ldots, 0, 2^{i_j}, 0, \ldots, 0)$, with $0 \leq$

$i_j \leq \log_2(N_j) + 1$. With these qubits, we can perform, as in the previous case, a Quantum Fourier Transform on each component to retrieve the secret.

As this approach does not use the split structure of the group to obtain any advantage, it is not expected to perform better than in the cyclic case. Moreover, with multiple independent components on which we need to obtain exact values, the algorithm may even perform worse.

We extend the 2-valuation to the vectors, and consider $\mathrm{val}_2(x_1, \ldots, x_m) = (a, b)$, with $a = \min_i x_i \neq 0$, $b = \mathrm{val}_2(x_a)$, with the lexicographic order. Intuitively, instead of sorting the elements by their divisibility by 2, we sort them sequentially, with their divisibility by 2 on the first component, then, if it is infinite, with the one on the second component, and so on.

We note δ_j^i the vector null on all components except the j-th one, where it is 2^i, and $n_j = \lceil \log_2(N_j) \rceil$.

Simulation results for products of $\mathbb{Z}/N\mathbb{Z}$ when $N \simeq 2^{10}$ and $N \simeq 2^{15}$ are given in Table 4. We try multiple settings, like having an increased number of elements in R for the first cyclic groups, or allowing to drop some elements in R instead of failing to obtain a 0. The changes in the cost seemed marginal at the scale we were able to simulate.

These simulations did not allow us to produce a precise estimate of the complexity. It seems to be still subexponential, but scales slightly worse. This may indicate that this approach may not be the best to tackle products of cyclic groups. There are still some settings that may be relevant, such as only computing the shift on one component at a time, or using the fact that we can choose in which order we tackle each group. Another approach would be to build up on [24, Algorithm 2], which produces elements in an increasingly smaller interval, instead of elements with an increasing 2-valuation.

m	$\log_2(Cost)$	$2.2\sqrt{m * 10} + 0.4$	$\|R\|$
1	7.6	7.4	2
2	10.4	10.2	3
3	12.5	12.4	3
4	14.3	14.3	3
5	16	16.0	3
6	17.4	17.4	3
7	18.8	18.8	3

(a) $\log_2(N) = 10$

m	$\log_2(Cost)$	$2.1\sqrt{m * 10} + 0.7$	$\|R\|$
1	8.9	8.8	3
2	12.3	12.2	3
3	14.8	14.8	3
4	17.0	17.0	3

(b) $\log_2(N) = 15$

Table 4: Simulation results for Algorithm 3, with 90% success

The change only becomes significant when there are many small groups. In the case of isogenies, the group is generally considered almost-cyclic. It has huge cyclic component, plus a few small ones. Moreover, the odd part is almost

Algorithm 3 Hidden shift algorithm for abelian groups

```
    Input: $N$, a number of queries $Q$, a quantum oracle access to $f$ and $g$ such that
    $f(x) = g(x + (s_1, \ldots, s_m)), x \in \bigotimes_{k=1}^{m} \mathbb{Z}/N_k\mathbb{Z}$
    Output: $(s_1, \ldots, s_m)$
 1: Generate $Q$ random labels in $\bigotimes_{k=1}^{m}[0; N_k)$ using the quantum oracles
 2: Separate them in pools $P_{j,i}$ of elements $e$ such that $\mathrm{val}_2(x) = (j, i)$
 3: $i \leftarrow 0$
 4: $R = \emptyset$
 5: for $j \in [1; p]$ do
 6:     while $\exists i : P_{j,i} \neq \emptyset$ do
 7:         if $i \leq n_j$ then                                  $\triangleright$ $n_j = \lceil \log_2(N_j) \rceil$
 8:             Pop a few elements $e$ from $P_{j,i}$, put $(e - \delta_j^i)$ in $R$.
 9:         end if
10:         for $e \in R$ do
11:             if $\mathrm{val}_2(e) = (j, i)$ then
12:                 Pop $a$ of $P_{j,i}$ which maximizes $\mathrm{val}_2(a + e)$ or $\mathrm{val}_2(e - a)$
13:                 $e = e \pm a$
14:             end if
15:         end for
16:         if $\{(\delta_j^i) | 0 \leq i \leq n, 1 \leq j \leq m\} \subset R$ then
17:             Apply a QFT on each component, measure a $t_j$
18:             $s_j \leftarrow \left\lceil \frac{-N_j t_j}{2^{n_j+1}} \right\rfloor \mod N_j$
19:             return $(s_1, \ldots, s_m)$
20:         end if
21:         while $|P_{j,i}| \geq 2$ do
22:             Pop two elements $(a, b)$ of $P_{j,i}$ which maximize $\mathrm{val}_2(a + b)$ or $\mathrm{val}_2(a - b)$
23:             $c = a \pm b$
24:             Insert $c$ in the corresponding $P_{j,i}$
25:         end while
26:         $i \leftarrow i + 1$
27:     end while
28: end for
29: return Failure
```

always cyclic [12], which means that the remaining groups in the product are small groups of size a power of 2, for which we can use the more efficient methods of [6]. Hence, for our estimates, we will consider that the group is cyclic.

5 Lattice Reduction

5.1 Finding a Basis of Multi-periods

The ideas in this section and the next one are fairly close to the ones in [3], where the authors use lattice reduction techniques to decompose arbitrary elements of $\mathcal{C}\ell(\mathcal{O})$ as products of small-degree isogenies (see Algorithm 7 in [3]). The main difference is that we are in a more precise setting, in which the ideal classes upon which we decompose elements of $\mathcal{C}\ell(\mathcal{O})$ are already given by construction on the scheme, and that we separate the computation of the small basis as precomputations.

Given p and the ideal classes $[\mathfrak{l}_1], \ldots, [\mathfrak{l}_u]$, we consider the set of integer vectors $\bar{e} = (e_1, \ldots e_u)$ such that $[\mathfrak{l}_1]^{e_1} \ldots [\mathfrak{l}_u]^{e_u} = \mathbf{1}$, or "multi-periods". This forms an integer lattice in $\mathbb{R}^u$, which we denote $\mathcal{L}$. This section and the following one are devoted to finding a *short* basis of it, but we first need to find a basis.

The complexity estimates in this section rely on the assumption that $\mathcal{L}$ behaves like a "random" lattice: in order to estimate the efficiency of lattice reduction algorithms, we apply heuristics found in the literature. It is likely that, if these heuristics do not apply for $\mathcal{L}$, then the lattice has more structure and more properties than expected, properties that could be exploited in other ways.

Computing Relations. In order to compute a short base of $\mathcal{L}$, one needs at least to input some vectors in the lattice. Finding those vectors, i.e, some multi-periods of $[\mathfrak{l}_1], \ldots, [\mathfrak{l}_u]$, is classically difficult. Quantumly, we cannot immediately apply Shor's period-finding algorithm [31], because the $[\mathfrak{l}_i]$ interfere with each other: it seems difficult to us to recover a lattice base using only period-finding in the $[\mathfrak{l}_i]$.

Instead, we use a generating set for the group spanned by $([\mathfrak{l}_1], \ldots, [\mathfrak{l}_g])$, with independent elements $([\mathfrak{g}_1], \ldots, [\mathfrak{g}_k])$ (this generating set is already needed by our application of Kuperberg's algorithm). Now we can use Shor's algorithm to decompose the $[\mathfrak{l}_i]$ on these generators, since they do not interfere (in other words, having $[\mathfrak{g}_1]^{j_1} \ldots [\mathfrak{g}_k]^{j_k} = \mathbf{1}$ implies $j_1 = 0 \mod ord([\mathfrak{g}_1])$, ..., $j_k = 0 \mod ord([\mathfrak{g}_k])$, while this is not the case for the $[\mathfrak{l}_i]$ themselves).

This decomposition enables us to quickly find some multi-periods, although they can be of high L_1 norm, using a polynomial amount of computations.

After these computations (polynomial in $\log(\#\mathcal{C}\ell(\mathcal{O}))$), we have a basis U of the lattice $\mathcal{L}$, which is full rank since it contains the vectors $(0 \ldots 0 \quad (\#\mathcal{C}\ell(\mathcal{O})) \quad 0 \ldots 0)$.

5.2 Finding a Short Basis

We now show that finding a short basis of $\mathcal{L}$ is a very easy step. Our estimations suggest that, once the first basis U is known (which is actually a more difficult step, as it involves Shor's algorithm), reducing it can be done in less than one hour using a simple classical processor, for all parameters considered here.

Since our goal is to output an approximate short basis, we use the best known algorithm to date, the Block Korkine Zolotarev algorithm (BKZ) [30]. Its complexity depends on the dimension u and the *block size*, an additional parameter which determines the quality of the basis. Practical complexity analyses are found in [21] and [9].

We do not study quantum lattice algorithms (see e.g [26]), since approximation algorithms are sufficient for us.

Remark 3 (Basis Quality Benchmarks). In works such as [9], the quality of the basis is related to the Hermite factor. Roughly speaking, the Hermite factor relates the L_2 norm of the shortest vector found with the u-th root of the lattice volume. The first vector of the basis B in output, b_1, is such that

$$||b_1||_2 \leq c^u (\mathrm{Vol}(L))^{1/u}$$

where c^u is the Hermite factor, and c a constant which depends on the algorithm used. For our purposes, it is better to work with the approximation factor, which relates $||b_1||_2$ and $\lambda_1(\mathcal{L})$, the euclidean norm of the smallest vector in $\mathcal{L}$. An approximation factor of c^{2u} is guaranteed, but in practice, it is equal to (and sometimes better than) the Hermite factor. So we consider:

$$||b_1||_2 \leq c^u \lambda_1(\mathcal{L}) \enspace .$$

From [21], we see that BKZ-20 gives us a heuristic constant c of approximately 1.0128. Furthermore, as noted in Section 3.1, this Hermite factor seems to be worst-case: this further justifies to consider that our lattice $\mathcal{L}$ behaves "heuristically". Furthermore, in [21, Fig. 12], the authors give a running time for BKZ-20 of the order 1000 CPU seconds for dimension 200. Hence we take this algorithm and $c = 1.0128$, ensuring that the small basis can be computed with negligible classical computations.

In the following, we are mostly interested in bounding $||b_1||_2$. We assumed above that there existed at least one vector $\bar{e} = (e_1 \dots e_u)$ with $e_i \in \{-m, \dots, m\}$ such that $\prod_i [\mathfrak{l}_i]^{e_i} = \mathbf{1}$. This only assumption suffices to write that $\lambda_1(\mathcal{L}) \leq 2m\sqrt{u}$, hence $||b_1||_2 \leq 2c^u m\sqrt{u}$.

Remark 4. There exists a classical subexponential algorithm to compute the class group [23]: Algorithm 4 could be performed classically in subexponential time (this is done by Couveignes in [13]). However, since we consider a quantum adversary, it makes sense to use quantum polynomial-time algorithms to compute the class group structure.

Algorithm 4 Finding a short basis of the lattice of multi-periods.

```
Input : The elements $([\mathfrak{l}_1], \ldots, [\mathfrak{l}_u])$, a generating set of the class group $([\mathfrak{g}_1], \ldots, [\mathfrak{g}_k])$
Output : A basis $B$ of the lattice $\mathcal{L}$ with (approximate) Hermite factor 1.0128.
1: Using Shor's period-finding algorithm, decompose the $[\mathfrak{l}_i]$ over the generating set given in input. From this decomposition, find vectors of $\mathcal{L}$ and compute a (non reduced) basis of the lattice.
2: Using BKZ-20 lattice reduction on a classical processor, find the short basis $B$.
3: return $B$
```

5.3 Solving the Approximate CVP with a Reduced Basis

Recall that we need to evaluate the group action for a product $\prod_i [\mathfrak{l}_i]^{t_i}$ for some t_i that can be large. This is where we use the lattice $\mathcal{L}$ and the short basis $B = b_1, \ldots b_u$ in output of Algorithm 4. Indeed, given a vector $\bar{v}$ in $\mathcal{L}$, we have:

$$\prod_i [\mathfrak{l}_i]^{t_i} = \prod_i [\mathfrak{l}_i]^{t_i - v_i} \ .$$

We are now interested in finding the vector $\bar{v}$ in $\mathcal{L}$ which is the closest possible to $\bar{t}$. The closest we can be, the less evaluating the group action will cost. This is an instance of the well-known lattice Closest Vector Problem, in our case the approximate CVP, since we are more interested in bounding the distance than obtaining the best possible vector.

Since the target vector is in superposition, this bound should hold for all vectors of $\mathbb{Z}^n$.

We use Babai's nearest-plane algorithm [1] (see e.g [19], chapter 18). Given the target vector $\bar{t}$, a reduced basis B and its Gram-Schmidt orthogonalization $B^\star$, this algorithm runs in polynomial time (more precisely, $O(u^2)$ operations, so this will be of no consequence in the complexity analyses below) and outputs a vector $\bar{v}$ in the lattice $\mathcal{L}$ such that:

$$||\bar{v} - \bar{t}||_2 \leq \frac{1}{2}\sqrt{\sum_{i=1}^{u} ||b_i^\star||_2^2}$$

where $B^\star$ is the Gram-Schmidt orthogonalization of B. In particular, we consider that $\sum_{i=1}^{u} ||b_i^\star||_2^2 \leq u\,||b_1||_2^2$ (we infer this from heuristics in [21] and [9] about the decreasing norms of the $b_i^\star$).

This gives:

$$||\bar{v} - \bar{t}||_2 \leq \frac{1}{2}\sqrt{u}\,||b_1||_2 \leq umc^u$$

where $c = 1.0128$. This bound holds simultaneously for every target vector $\bar{t}$ and corresponding output $\bar{v}$ by Babai's method.

Effect on the L_1 Norm. We are interested on the L_1 norm of the difference $\bar{v} - \bar{t}$. Indeed, we count an evaluation of the group action for $\prod_i [\mathfrak{l}_i]^{t_i - v_i}$ as $||\bar{v} - \bar{t}||_1 / (um)$ equivalent classical evaluations. The closest we are to the lattice $\mathcal{L}$, the smallest the representation (via $\bar{v} - \bar{t}$) of class group elements becomes; the closer we are to a class group action evaluation with all exponents in $\{-m, \dots, m\}$.

We have:

$$||\bar{v} - \bar{t}||_1 \leq \sqrt{u}\, ||\bar{v} - \bar{t}||_2 \leq u^{3/2} m c^u \ .$$

The multiplicative factor w.r.t the classical group action (mu) is $u^{1/2} c^u$.

Algorithm 5 Finding a short representation of an element of the class group, over the $[\mathfrak{l}_i]$.

```
   Input : A vector $\bar{t}$ representing an element of the class group of the form $\prod_i [\mathfrak{l}_i]^{t_i}$,
   a basis $B$ for the lattice $\mathcal{L}$ given by Algorithm 4 with Hermite factor $c$.
   Output : A vector $\bar{s}$ such that $||s||_1 \leq u^{3/2} m c^u$ and $\prod_i [\mathfrak{l}_i]^{t_i} = \prod_i [\mathfrak{l}_i]^{s_i}$.
1: Using Babai's nearest-plane method with the basis $B$, find $\bar{v}$ in $\mathcal{L}$ such that
   $||\bar{t} - \bar{v}||_1 \leq u^{3/2} m c^u$.
2: return $\bar{t} - \bar{v}$.
```

Algorithm 4 and Algorithm 5 above are the two main components of Algorithm 7 in [3] for faster isogeny evaluations. The authors also use BKZ to reduce the lattice base and Babai's algorithm to solve the approximate CVP instance. They however consider the general asymptotic ordinary case, for which an interesting basis is not given to the attacker, and the classical case, while we separated the two algorithms to reduce the quantum costs.

In [4], this algorithm for fast evaluation is further applied to the CSIDH scheme. However, the authors are more focusing on the asymptotic time complexity. It can be remarked that asymptotically, using the ideals $[\mathfrak{l}_i]$ provided by the scheme to decompose any ideal is not the best method. Indeed, the multiplicative factor guaranteed by BKZ increases exponentially in the dimension u ; one can try to increase the block size of BKZ in order to reduce at best the complexity of the oracle evaluation, but this happens to be always (asymptotically) slower than taking a basis with a limited number of ideals (less than the given u) and greater exponents than the given m.

We are now able to give the complexity for all three NIST levels given in [7], as guaranteed by heuristics (Table 6). We remark that computing the action of $[\mathfrak{g}]$ actually requires three steps:

- Decomposing $[\mathfrak{g}]$ over the $[\mathfrak{l}_i]$, as $[\mathfrak{g}] = \prod_i [\mathfrak{l}_i]^{t_i}$. This is done using Shor's algorithm, as before;
- Using Babai's nearest-plane algorithm with the basis B, finding the approximate closest vector $\bar{v}$: this requires u^2 multiplications of vectors coordinates, which are approximately $\log_2 p$-bit integers;

- Computing the action of $\prod_i [\mathfrak{l}_i]^{t_i - v_i}$.

For each set of parameters, the last step is the most costly of the three. (For example, for $u = 200$, Babai's method costs $u^2 = 4 \cdot 10^4$ $\log_2 p$-bit multiplications, while the action costs approximately $3.96 \cdot 10^5$ small isogeny evaluations). This is why the time complexity of the quantum group action oracle depends exclusively on its overhead w.r.t the classical one (additional computations are negligible).

Table 5: Guaranteed cost overhead on the computation of the group action, w.r.t. a legitimate key exchange computation.

Level	$\log_2 p$	u	m	Overhead
NIST 1	512	74	5	$\leq 2^5$
NIST 3	1024	132	7	$\leq 2^6$
NIST 5	1792	209	10	$\leq 2^8$

Simulation Results. Most of the cases, the odd part of the class group $\mathcal{C}\ell(\mathcal{O})$ is cyclic, as shown by the Cohen–Lenstra heuristics [12]. We took cardinalities q at random and performed computations either with:

- Random lattices $\mathcal{L}$ corresponding to a cyclic group with cardinality q, taking some u elements at random in this group and computing two-by-two relations between them;
- Lattices $\mathcal{L}$ generated by $u - 1$ random vectors from $\mathbb{Z}[q]^u$ and the vectors of the form $(0 \dots q \dots 0)$.

The results happen to be the same in both cases. Thus, our heuristics below hold as long as we are able to find $u - 1$ independent relations, which is trivially possible if the group is cyclic. If this is not the case, there are additional factors.

When given a lattice of the first family, the computational system Sage [33] performs BKZ reduction with blocksize 20 in a handful of minutes in dimension 200 (this seems to come from the fact that the basis is, in that case, very sparse). With the second family, the time greatly increases but remains manageable with a single PC.

In order to bound the L_1 norm of the vector in output of Babai's algorithm, we are interested in the quantity:

$$A = \sqrt{u} \frac{1}{2} \sqrt{\sum_{i=1}^{u} ||b_i^\star||_2^2}$$

where $B^\star$ is the Gram-Schmidt orthogonalization of the output basis B. We compute this quantity, on average, for a handful of lattices with dimensions

Table 6: Simulation cost overhead on the computation of the group action, w.r.t. a legitimate key exchange computation.

Level	$\frac{1}{2}\log_2 p$	u	m	A	Factor
NIST 1	256	74	5	$\simeq 550$	$\leq 2^2$
NIST 3	512	132	7	$\simeq 2300$	$\leq 2^2$
NIST 5	896	209	10	$\simeq 9500$	$\leq 2^3$

those of the CSIDH parameters we are interested in. The deviation from the values found does not exceed 10%.

As a consequence, we only count the oracle overhead as a (small) factor 2^3 in what follows.

Remark 5. Once the secret isogeny has been recovered, the factors in Table 6 also give the multiplicative overhead on the size of the representation found w.r.t the original one.

6 Consequences on Ordinary Curves

Although we focused until now on the specific case of CSIDH, lattice reduction can be applied to the general ordinary isogeny problem, exactly in the lines of Couveignes [13], Rostovtsev and Stolbunov. Indeed, if ℓ is an Elkies prime (approximately a half of them), it gives rise to two prime ideals of norm ℓ and corresponding isogenies of degree ℓ. Given enough such primes (say u), it is possible to span the whole group $\mathcal{C}\ell(\mathcal{O})$ with products of these ideals. For CSIDH, we needed powers in $\{-m, \dots, m\}$ with $2m+1 \simeq p^{1/(2u)}$, for u was restricted by the parameter p. In general, we do not have such a restriction.

Original CRS Scheme. The original Couveignes–Rostovtsev–Stolbunov scheme computes the class group action using these Elkies primes. The difference with CSIDH comes from the fact that the trace of the Frobenius is not 0, so computing one of the small isogenies is actually computationally costly. In particular, contrary to the case chosen in the CSIDH scheme, where ℓ splits as $(\ell, \pi-1)$ and $(\ell, \pi+1)$, finding the two neighbors in the isogeny graph requires to compute the roots of a modular polynomial of degree $\ell+1$, costing $\widetilde{O}(\ell \log p)$ operations in $\mathbb{F}_p$. Hence, while Elkies ideals are easily found, computing their action is much slower.

Quantumly, we can use lattice reduction the same way as we do for CSIDH, with adaptations. The multiplicative factor of computing the group action in superposition over all elements of $\mathcal{C}\ell(\mathcal{O})$ depends on the dimension u as $u^{1/2}c^u$, where c is the Hermite factor of the lattice basis reduction algorithm used. However, in the CRS scheme, u should be the greatest possible and m the smallest in order to speed up the computations. There is no limit on u, since one can find as many Elkies primes as needed. So we may take $m = 1$ and $u = \frac{1}{2}\frac{\log p}{\log 3}$. This gives a set of ideal classes in $\mathcal{C}\ell(\mathcal{O})$: $[\mathfrak{l}_1], \dots, [\mathfrak{l}_u]$ such that each element of $\mathcal{C}\ell(\mathcal{O})$ is of the form:

$$[\mathfrak{g}] = [\mathfrak{l}_1]^{e_1} \cdots [\mathfrak{l}_u]^{e_u}$$

with $e_i \in \{-1, 0, 1\}$.

If we take p a 512-bit prime, the lattice dimension becomes as high as 162. With $c = 1.0128$, we obtain a multiplicative overhead of $u^{1/2} c^u \simeq 98 \leq 2^7$: evaluating the group action in superposition could cost almost a hundred times more than the classical. With $\log_2 p = 1024$, we get a dimension 323 and a factor 2^{10}, with $\log_2 p = 1792$ we have a dimension 565 and a factor 2^{15}. To reduce these complexities, we may want to reduce the Hermite factor, but since the dimension has increased, this will turn out to be difficult. The time complexity of BKZ-20, that we advocated above, increases exponentially. We extrapolate from [21] a potential cost of 2^{60} computations in dimension 565. The cost is almost balanced with the second step and it seems difficult to improve it significantly.

Experimental Results. As with the supersingular case, we performed experiments with the lattice of multi-periods of random elements in a random cyclic group. The relations by which we generate the lattice being extremely simple, the computations are much faster than expected, and the results are far better than our upper bounds. In dimension 162, the number of isogenies to be evaluated by the quantum oracle is approximately 800 (with 10% deviation), which represents a multiplicative overhead of approx. $5 \leq 2^3$, since the classical group action needs 162 isogenies (the precise dimension of the lattice). In dimension 323, we get approximately 3500 isogenies and an overhead $10 \leq 2^4$.

Asymptotic Cost. In order to have a guaranteed asymptotically efficient algorithm, one should apply in this case the general method given in [3] (Algorithm 7). The basis taken has dimension of the order $O(\log^{2/3}(\sqrt{p}))$ instead of the maximal possible, in order to control the cost of BKZ in the basis reduction step. The use of quantum algorithms to perform this step would have direct consequences on the asymptotic complexity of this method.

Hybrid Approach in [18]. In [18], the authors consider a hybrid approach: $\mathcal{C}\ell(\mathcal{O})$ is generated by a set of ideals among which some are baseline Elkies, while some are evaluated faster, due to their special properties. The resulting isogeny walks are divided into "Elkies steps" for the former ones and "Vélu steps" for the latter ones. The goal is then to perform the most Vélu steps possible; Elkies steps are still necessary to ensure a key space of sufficient size. Furthermore, we need to be cautious with some Vélu ideals, that can only be used in one direction.[5] This turns out to be an optimization problem, which gives bounds on the exponents depending on the ideals. Hence $\mathcal{C}\ell(\mathcal{O})$ is now spanned by products of the form:

$$[\mathfrak{l}_1]^{e_1} \cdots [\mathfrak{l}_u]^{e_u} [\mathfrak{l}_{u+1}]^{e_{u+1}} \cdots [\mathfrak{l}_{u+v}]^{e_{u+v}}$$

[5] Since the trace of the Frobenius is 0 for supersingular curves, all steps in CSIDH turn out to be Vélu steps: this is where the scheme gains its efficiency.

where the powers of $[\mathfrak{l}_i], 1 \leq i \leq u$ go from $-m_i$ to m_i and the powers of the $[\mathfrak{l}_i], u < i \leq u+v$ go from 0 to m_i (only one direction should be used).

More Precomputations. A solution to the approximate CVP given by Babai's algorithm cannot be automatically forced to have some of its coordinates positive. To overcome this issue later, we consider the set S of vectors $\bar{f}$ with $u+v$ coordinates such that:

$$[\mathfrak{l}_1]^{f_1} \cdots [\mathfrak{l}_u]^{f_u} [\mathfrak{l}_{u+1}]^{f_{u+1}} \cdots [\mathfrak{l}_{u+v}]^{f_{u+v}} = [\mathfrak{l}_{u+1}]^{-n_1} \cdots [\mathfrak{l}_{u+v}]^{-n_v}$$

where $n_1, \dots n_v$ will be chosen later. Computing this set is easy. We can find a vector $\bar{f}_0$ in S using Shor's algorithm, then $S = \bar{f}_0 + \mathcal{L}$. Finding an approximate shortest vector in S is an instance of the closest vector problem, since we look for the vector in $\mathcal{L}$ closest to $\bar{f}_0$.

Remark 6. It is a crucial point that the m_i are different. Hence, we may want to use lattice algorithms with a weighted norm:

$$||e_1 \dots e_{u+v}||^2 = \left(\frac{e_1}{m_1}\right)^2 + \dots \left(\frac{e_{u+v}}{m_{u+v}}\right)^2 \ .$$

Alternatively, we consider the lattice $\mathcal{LM}$ obtaining by weighting the coordinates e_i by m_i. We apply standard lattice reduction algorithms to $\mathcal{LM}$. Given a vector $\bar{e}$ in $\mathcal{L}$, the multiplicative factor between the quantum group action oracle and a classical key exchange can be bounded by $\max \left|\frac{e_i}{m_i}\right|$. Hence, we are interested in L_∞ norms of elements of $\mathcal{LM}$.

The norm of the shortest vector in the reduced basis of $\mathcal{LM}$ is bounded by:

$$||b_1||_2 \leq c^{u+v} \lambda_1(\mathcal{LM}) \leq 2c^{u+v}\sqrt{u+v} \ .$$

And the shortest (weighted) vector $\bar{f}$ in S (from the analysis in Section 5.2) has norm:

$$||\bar{f}||_2 \leq (u+v)c^{u+v} \ .$$

We can bound the absolute value of each of its coordinates by $(u+v)c^{u+v}$.

Now that these precomputations are done, we can solve the approximate CVP instance for a given $\bar{s}$.

CVP with Some Positive Coordinates. Given a (weighted) vector $\bar{s}$, Babai's algorithm returns a vector $\bar{t} = (t_1, \dots t_u, t_{u+1}, \dots t_{u+v})$ of $\mathcal{LM}$ such that:

$$||\bar{s} - \bar{t}||_2 \leq c^{u+v}(u+v) \ .$$

The problem is that the last v coordinates of $\bar{s} - \bar{t}$ should all be positive, in order to compute efficiently the group action.

To ensure that, we now take the (weighted) vector $\bar{f}$ of above with $n_1, \dots, n_v$ all equal to $2c^{u+v}(u+v)$.

This ensures that: $s_{u+1} - t_{u+1} + f_{u+1} + n_1 \geq 0$, and so on.

We output the vector with (weighted) coordinates $s_i - t_i + f_i$ for $i \leq u$ and $s_i - t_i + f_i + n_i$ for $u < i \leq u + v$. (It suffices to multiply the i-th coordinate by m_i to obtain the corresponding actual integer values of the exponents). Its L_∞ norm can be bounded by $||\bar{s} - \bar{t}||_2 + ||\bar{f}||_2 + n_1 \leq 4(u + v)c^{u+v}$, giving the multiplicative factor w.r.t the classical group action.

Consequences. In the following, we perform computations with the example given in [18], Section 6. There are 13 Vélu primes used in two directions, 11 in one direction, 29 Elkies primes. The total dimension is $u + v = 53$. It is relatively small, so that exact lattice methods could even be performed in order to speed up the computations (we should be able to solve the exact shortest vector problem in $\mathcal{LM}$). With $c = 1.0128$, we estimate $4(u+v)c^{u+v} \simeq 416 \leq 2^9$. Although decomposing the class group and ideal classes given in [18] seems not computationally feasible, we performed some experiments with the same dimension and weights, but on a random lattice from a cyclic group as before. The results suggest that this factor could actually be lower than 2^5 (we compute it as four times the quantity $\frac{1}{2}\sqrt{\sum ||b_i^\star||^2}$ for the reduced basis, which is the bound on the euclidean norm given by Babai's algorithm).

If we computed instead the class group action using a maximal basis of Elkies primes, dismissing the refined structure from [18], we would obtain a dimension 162 in this example (with a 511-bit prime). Our experimental estimates give a multiplicative overhead of 2^3 in that case, but the cost of evaluating the group action would be at least 4 times greater than using the refined structure (we estimate even 12). In the end, and since our cost analysis above was very approximate, the advantage goes to the refined group action computation.

If we add the cost of the hidden shift, we obtain a quantum cost of at most 2^{37} time and 2^{31} memory. We do not generalize this cost, as unlike for CSIDH, we do not see a clear asymptotical behaviour for the parameters of the scheme.

This seems to be a general principle: whenever there is a way to classically speed up the class group action using a refined basis, the lattice reduction step can be adapted to use it as well in the quantum oracle. Besides, the latter can benefit from a reduced dimension.

7 Complexities and NIST Benchmarks

The parameters in [7] are aimed at three NIST security levels, defined as follows in the post-quantum public-key cryptography standardization process [28]:

- NIST 1: breaking the scheme requires as much resources as a quantum key-recovery on AES-128;
- NIST 3: as hard as a quantum key-recovery on AES-192;
- NIST 5: as hard as a quantum key-recovery on AES-256.

A key-recovery on AES is done using Grover's algorithm, which runs in approximately $2^{|k|/2}$ iterations, where $|k|$ is the length of the key, each iteration requiring one or more evaluations of a quantum circuit implementing AES. Such a circuit was designed in [22]. It costs approximately 2^{20} quantum gates, when counting on the universal Clifford+T set (we will use this same set for all quantum circuits below).

Quantum Circuits for the CSIDH Group Action. In the first version of this paper, we gave an estimation of the quantum gate cost required to evaluate the CSIDH group action and, by consequence, to attack the scheme using Kuperberg's algorithm. Since then, more precise estimations have been given in [2]. We can now base our analysis on these results. The authors of [2] give 0.7×2^{40} nonlinear bit operations for the CSIDH-512 instance, in order to reach a success probability of the order 2^{-32}, necessary since we require that much queries. This cost comes mainly from the $2^{21.4}$ multiplications in $\mathbb{F}_p$ needed, each one costing $2^{18.7}$ Toffoli gates, with $\log_2 p = 512$. The number of T gates is $2^{43.3}$. The total number of gates (Clifford + T) is of the order $2^{45.3}$. Furthermore, in order to keep the number of ancilla qubits sufficiently low, some inner levels of uncomputation are needed, possibly increasing the computation by some factor (at least 4). In the end, the quantum CSIDH group action oracle for a prime of 512 bits should cost, in our setting, approx. 2^{48} Clifford+T gates, instead of the 2^{37}.

This cost is not given in [2] for other values of p, but we can roughly estimate the increasing number of multiplications to be performed (counting the increasing dimension, the increasing value of the little primes and the increasing precision needed). Furthermore, when p is doubled, the cost of a multiplication at most quadruples. For CSIDH-1024, we can take 2^{53} gates and 2^{56} for CSIDH-1792. Notice that from the point of view of depth, the oracle contains almost all the depth of the whole circuit (due to the structure of Kuperberg's algorithm).

Counting in Classical Group Actions. If we count in multiples of the quantum CSIDH group action evaluation, and do not go into the quantum implementation details, the cost of the attack is given in Table 7.

Table 7: Attacking CSIDH. Complexities are given in $\log_2$ scale, in number of key exchanges.

Level	Memory	Requests	Appr. factor	**Total cost**	Expected
NIST 1	31	32.5	2	**34.5**	62
NIST 3	43	44.5	2	**46.5**	94
NIST 5	56	57.5	3	**60.5**	129

Counting in Quantum Gates. The benchmarks advocated in the NIST call [28] do not count the complexity of an attack comparatively to that of the scheme, but instead, completely rely on a comparison with attacks on AES variants. In order to check whether the security levels are met, we compare the complexity in Clifford+T gates with the AES instance corresponding to the targeted level (given in [22]; we reduce the gate counts by supposing that we use only respectively one, two and two plaintext-ciphertext pairs, since this seems enough to guarantee a good success probability).

Table 8: CSIDH attack *time* complexities in $\log_2$ scale, compared with the corresponding Grover key-recovery on AES.

Level	AES instance attacked	Time cost of the corresponding Grover	Total time cost of our attack	Previous $\log_2 p$	Updated $\log_2 p$
NIST 1	AES-128	85.9	82.5	512	600
NIST 3	AES-192	119.1	99.5	1024	1900
NIST 5	AES-256	151.3	114.5	1792	4100

Table 8 shows that the parameters proposed in [7] all fall below the targeted security levels, albeit by a short margin for the 512-bit proposal. This mainly comes from the cost of quantumly evaluating CSIDH. The oracle accounts for more than half of the exponent in all cases.

We provide indicative values of $\log_2 p$ that should make our attack cost more than the reference AES key-recovery.

With the MAXDEPTH *Assumption.* The NIST document suggests to assess the security against quantum attackers when they are restricted to "a fixed running time, or circuit depth" [28]. To this respect, there is a significant difference between Kuperberg's and Grover's algorithms.

Kuperberg's circuit, forming a tree of combinations between qubits, is actually very "flat": the total amount of quantum gates applied to a given qubit is bounded by that of the group action oracle, with a negligible overhead due to the combinations. Implementations of this algorithm would likely use this specific structure at their advantage.

On the contrary, Grover's algorithm performs a long, sequential computation. Restricting the depth means restricting the number of iterations. When dividing the number of iterations by S, the probability of success is roughly divided by S^2. It is well-known [34] that this square factor cannot be overcome. In total, this induces a significant overhead on the quantum running time, since we run S^2 instances with T/S gates each instead of a single instance with T gates.

Introducing the parameter MAXDEPTH, we suppose that only a limited number of operations can be performed on a given qubit. This does not mean that the *whole* quantum computation stops after MAXDEPTH operations. In a first step, the group action oracle is applied $2^{O(\sqrt{n})}$ times, but to separate qubits. Only then are combinations performed; a given qubit is used in at most a logarithmic number of them (negligible depth). Therefore, as long as MAXDEPTH is greater than the oracle depth, the quantum time of this attack is not affected. Otherwise, one would need to reduce the depth of $\mathbb{F}_p$-multiplications. The T-depth of AES quantum oracles is given in [22] as approximately 2^{17}. In both cases, one would trade off improvements in depth for increased time and memory complexities.

Taking MAXDEPTH between 2^{60} and 2^{96} gives an advantage to the adversary, in that Grover's time complexity is increased while the attack complexity remains the same. This counterintuitive conclusion actually accounts for the advantage, for a quantum adversary, to be able to run efficiently a massively parallel computation instead of a long, serial (and badly parallelized) one. In the case of CSIDH, it appears much more meaningful to drop the MAXDEPTH assumption and set it to $+\infty$.

8 Conclusion

We presented a comprehensive quantum security assessment of CSIDH. In particular, when compared to the cost of a classical key-exchange, we showed that the parameters set in [7] actually seem to provide only around half of the expected security, as summarized in Table 7.

Protecting CSIDH Against this Attack. In Section 7, we provided extensive complexity estimates of this attack on CSIDH, depending on the benchmark. In particular, Kuperberg's algorithm as a quantum circuit has a much lower depth than Grover's, a particularity that may help to overcome possible limitations on running large serial quantum computations. We also reach the limits of the NIST benchmarking: by forcing the quantum circuits to have a maximal depth, in order to limit the capacities of a quantum adversary, we actually give an advantage to Kuperberg's algorithm against Grover. Reaching the same security levels would then require much higher parameters.

If we consider a gap and not the NIST benchmark, the base field size should by multiplied by around 4, that is, for example a p of size around 2048 would induce a gap of 2^{64} between the attack and the key exchange.

Other Isogeny-based Schemes. The NIST candidate SIKE [17] is not affected by this attack, as it uses supersingular elliptic curves on $\mathbb{F}_{p^2}$.

We stated above that lattice reduction could be used in the quantum cryptanalysis of the original Couveignes–Rostovtsev–Stolbunov scheme on ordinary curves, as remarked by multiple authors [18, 7, 3], by setting up a quantum class group action oracle alternative to the one given in [11], and we provided explicit cost estimates.

Recently, De Feo, Kieffer and Smith [18] proposed to refine the CRS scheme for ordinary curves in order to make it more practical, while maintaining the same level of security against a quantum adversary. Their approach can be seen as a hybrid between CRS and CSIDH. We showed that lattice techniques could be adapted to such a situation, in order to achieve a better time complexity for the quantum class group action oracle.

The recent signature scheme proposal SeaSign [16] uses the same security assumptions as CSIDH, and is just as much affected.

Other Quantum Algorithms. We based the quantum attack of this paper on an algorithm whose complexity is close to the one of [6]. The algorithm presented in [25] may be more efficient in our cases, which would improve this attack.

Our estimates show that when attacking an isogeny-based primitive using the commutative action of $\mathcal{C}\ell(\mathcal{O})$, such as CRS or CSIDH, the superposition oracle for computing the action of $\mathcal{C}\ell(\mathcal{O})$ based on the ideals used by the scheme itself is practical. Although its asymptotic time complexity is not optimal, in practice, lattice reduction steps can be performed efficiently and make the overhead small w.r.t the classical group action. In our experiments, this factor was reduced below 2^3 and we believe that a conservative security estimate should take it equal to 1. Moreover, as very little is known on the explicit complexity of the hidden shift problem in the general case, we consider that a conservative approach should take into account the most time-efficient algorithm known.

Acknowledgements. The authors want to thank María Naya-Plasencia for her helpful comments, Alain Couvreur and Jean-Pierre Tillich for helpful discussions on isogeny-based cryptography, Lorenz Panny and Joost Renes for their valuable comments on a draft of this paper. Thanks to Jean-François Biasse for pointing out the reference [3], Luca De Feo, Ben Smith and Steven Galbraith for helpful comments on Kuperberg's algorithm and discussions on the NIST benchmark.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n^o 714294 - acronym QUASYModo).

References

1. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986), https://doi.org/10.1007/BF02579403
2. Bernstein, D.J., Lange, T., Martindale, C., Panny, L.: Quantum circuits for the csidh: optimizing quantum evaluation of isogenies. Cryptology ePrint Archive, Report 2018/1059 (2018), https://quantum.isogeny.org, https://eprint.iacr.org/2018/1059
3. Biasse, J.F., 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)
4. Biasse, J.F., Jacobson, M.J., Iezzi, A.: A note on the security of CSIDH. CoRR (2018), https://arxiv.org/abs/1806.03656
5. Biasse, J., Jao, D., Sankar, A.: A quantum algorithm for computing isogenies between supersingular elliptic curves. In: Meier, W., Mukhopadhyay, D. (eds.) Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings. Lecture Notes in Computer Science, vol. 8885, pp. 428–442. Springer (2014)

6. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. Cryptology ePrint Archive, Report 2018/432 (2018), https://eprint.iacr.org/2018/432
7. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. To appear in: Advances in Cryptology - ASIACRYPT 2018 - 24th Annual International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, Australia, December 02-06, 2018
8. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. Cryptology ePrint Archive, Report 2018/383 (2018), https://eprint.iacr.org/2018/383
9. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 1–20. Springer (2011)
10. Cheung, K.K.H., Mosca, M.: Decomposing finite abelian groups. Quantum Information & Computation 1(3), 26–32 (2001), http://portal.acm.org/citation.cfm?id=2011341
11. Childs, A.M., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. J. Mathematical Cryptology 8(1), 1–29 (2014), https://doi.org/10.1515/jmc-2012-0016
12. Cohen, H., Lenstra, H.W.: Heuristics on class groups of number fields. In: Number Theory Noordwijkerhout 1983, pp. 33–62. Springer (1984)
13. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
14. Delfs, C., Galbraith, S.D.: Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. Des. Codes Cryptography 78(2), 425–440 (2016), https://doi.org/10.1007/s10623-014-0010-1
15. Ettinger, M., Høyer, P.: On quantum algorithms for noncommutative hidden subgroups. In: Meinel, C., Tison, S. (eds.) STACS 99. pp. 478–487. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
16. Feo, L.D., Galbraith, S.D.: Seasign: Compact isogeny signatures from class group actions. Cryptology ePrint Archive, Report 2018/824 (2018), https://eprint.iacr.org/2018/824
17. Feo, L.D., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Mathematical Cryptology 8(3), 209–247 (2014), https://doi.org/10.1515/jmc-2012-0015
18. Feo, L.D., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. Cryptology ePrint Archive, Report 2018/485 (2018), https://eprint.iacr.org/2018/485
19. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press (2012), https://www.math.auckland.ac.nz/ sgal018/crypto-book/crypto-book.html
20. Galbraith, S.D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. IACR Cryptology ePrint Archive 2017, 774 (2017), http://eprint.iacr.org/2017/774
21. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey,

April 13-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4965, pp. 31–51. Springer (2008)
22. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (ed.) Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9606, pp. 29–43. Springer (2016)
23. 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)
24. Kuperberg, G.: A Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. SIAM J. Comput. 35(1), 170–188 (2005), http://dx.doi.org/10.1137/S0097539703436345; http://dblp.uni-trier.de/rec/bib/journals/siamcomp/Kuperberg05
25. Kuperberg, G.: Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. In: 8th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2013, May 21-23, 2013, Guelph, Canada. pp. 20–34 (2013), http://dx.doi.org/10.4230/LIPIcs.TQC.2013.20
26. Laarhoven, T., Mosca, M., van de Pol, J.: Finding shortest lattice vectors faster using quantum search. Des. Codes Cryptography 77(2-3), 375–400 (2015), https://doi.org/10.1007/s10623-015-0067-5
27. Meyer, M., Reith, S.: A faster way to the csidh. Cryptology ePrint Archive, Report 2018/782 (2018), https://eprint.iacr.org/2018/782
28. NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf
29. Regev, O.: A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space. CoRR (2004), http://arxiv.org/abs/quant-ph/0406151
30. Schnorr, C., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Program. 66, 181–199 (1994), https://doi.org/10.1007/BF01581144
31. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 124–134. IEEE Computer Society (1994), http://dx.doi.org/10.1109/SFCS.1994.365700
32. Stolbunov, A.: Cryptographic schemes based on isogenies (2012)
33. The Sage Developers: SageMath, the Sage Mathematics Software System, `http://www.sagemath.org`
34. Zalka, C.: Grover's quantum searching algorithm is optimal. Physical Review A 60(4), 2746 (1999)