Quantum Security Analysis of CSIDH and Ordinary Isogeny-based Schemes

Xavier Bonnetain and André Schrottenloher

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

Abstract. CSIDH is a recent proposal by Castryck, Lange, Martindale, Panny and Renes for post-quantum non-interactive key-exchange. 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 show that those security parameters were too optimistic. Our result relies on two steps: first, we give a more precise complexity analysis of the hidden shift algorithm in this context, which greatly reduces the number of group actions to compute; second, we show how to compute efficiently this group action.
Our computations show, for example, that the 128-bit classical, 64-bit quantum security parameters proposed actually offer at most 37 bits of quantum security.
Finally, we extend our analysis to ordinary isogeny computations, and show that an instance proposed by De Feo, Kieffer and Smith and expected to offer 56 bits of quantum security offers at most 41 bits of quantum security.

Keywords: Post-quantum cryptography, isogeny-based cryptography, 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 [23] 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.

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 [8]. It runs in *subexponential* time, albeit not polynomial like Shor's algorithm, while all known classical attacks require exponential time. Informally, 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 [18].

As a consequence, there has been a move towards isogeny-based cryptography using supersingular elliptic curves [11]. 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 [14] for a survey).

CSIDH. CSIDH is a new asymmetric primitive proposed in [5] 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 [8] still applies. This is taken into account by the authors of [5]. 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 $N = O(\sqrt{p})$ and p is the prime number used for the base field, and this complexity represents the number of evaluations of the group action required to break the scheme.

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 [5] 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 competition: 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.

Quantumly Attacking CSIDH. In this paper, we show that although the parameters proposed initially in [5] seem compliant with the asymptotic complexity of

the currently known quantum attack, they all offer less quantum security than claimed, respectively less than 34, 47 and 62 bits of security instead of 62, 94 and 129.

This is a consequence of two crucial points: first, the most time-efficient version of Kuperberg's algorithm has a better complexity than estimated in [5]. This would be of less consequence if the group action evaluated in Kuperberg's algorithm turned out to be costly. Indeed, in the attack on ordinary curves of [8], much work is put into evaluating the group action in subexponential time, which is not trivial, since one needs to represent all isogenies in a sufficiently small way.

Second, we show that evaluating efficiently the commutative group action adds only little overhead, using lattice reduction methods that date back to Couveignes [9], were used in [2] in order to compute efficiently large-degree isogenies and were already mentioned in [5] (Section 7.2, "subexponential vs. practical") as a potential improvement, although not studied precisely.

Our attack complexities, for each set of parameters, are given in multiples of the cost of a key-exchange using CSIDH.

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 [12].

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 [5].

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 [10], where we find it first exploited to compute isogenies between supersingular elliptic curves.

In [3], 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 $\mathcal{Cl}(\mathcal{O})$. Moreover, a rational ℓ-isogeny from such a curve corresponds to an ideal of norm ℓ in $\mathcal{Cl}(\mathcal{O})$. The (commutative) class group $\mathcal{Cl}(\mathcal{O})$ 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 [8], 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*.[3] 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 [5]:

Proposition 1 (Prop. 8 in [5]). *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 $\mathcal{O}$.

[3] This is also the definition of a *hard homogeneous space* by Couveignes [9].

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

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

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 [5], $m = 5$. 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 [5] in Table 1.

Table 1. Approximate parameters for the three security levels of [5].

Level	Expected quantum security	$\log_2 p$	u	m
NIST 1	64	512	74	5
NIST 3	96	1024	128*	8*
NIST 5	128	1792	200*	11*

*In [5], 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 $[\mathfrak{b}]$. Using Velu's formulae, the evaluation of an isogeny of degree ℓ would cost $O(\ell)$ operations. The authors of [5] provide (Section 8) an efficient implementation of the class group action for one of the $\mathfrak{l}_i$, and $\sum_{i=1}^{u} |e_i|$ such small isogenies are to be applied (instead of $[\mathfrak{b}]$, which could be of high degree).

2.3 Claimed Security of CSIDH

We review the security claims of CSIDH in [5]. 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 [8], 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 [8] 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 [5]. 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 [5], 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 [5], [9] and [2]:

- Kuperberg's algorithm, which we use, was already considered in [5], Section 7.2 (quantum security), but the quantum complexity analysis was only partial.
- Using lattice reduction to compute efficiently the group action was mentioned in [5] as a possible method, and already done in [9] in a very similar setting, albeit a classical one. In [2], lattice reduction techniques are used to obtain short representations of large-degree isogenies as elements of $\mathcal{C}\ell(\mathcal{O})$.

Our 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 $\vec{f} = f_1, \dots f_u$ 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 Details of the Attack

In this section, we provide a more thorough analysis of the two phases of our attack and more precise computations.

4.1 Solving the Hidden Shift Problem with Kuperberg's Algorithm

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

Problem 1 (Hidden shift problem). Let $(\mathbb{G}, +)$ 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 [5], 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 variants of Kuperberg's algorithm to solve the hidden shift problem in commutative groups. The first algorithm was proposed by Kuperberg in [18], where it is shown to have a complexity in quantum query and memory of $\widetilde{O}(2^{\sqrt{2\log_2(3)\log_2(N)}})$ for the group $\mathbb{Z}/N\mathbb{Z}$. 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 [21], which has been generalized by Kuperberg in [19], 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 [8], with the same complexity.

Concrete Estimates. The $\widetilde{O}$ are not practical for concrete estimates. However, in [4], 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 the cyclic case seems to be the worst case, we will consider a quantum query and memory complexity of $2^{\sqrt{2\log_2(3)n}}$ for a group of 2^n elements. A sketch of reduction from $\mathbb{Z}/N\mathbb{Z}$ to $\mathbb{Z}/2^n\mathbb{Z}$ is proposed in the next section.

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 tradeoff algorithms between the time-efficient and memory-efficient variants [19].

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 qubits decohere, we can just drop them and continue 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 [7], as done in [8]. We are only interested in the subgroup spanned by $([\mathfrak{l}_1], \ldots, [\mathfrak{l}_u])$ (which should be close to the total group): we can get it using the same method. We then obtain a generating set $([\mathfrak{g}_1], \ldots, [\mathfrak{g}_g])$.

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

Efficiently Applying Kuperberg's Algorithm. The idea is to use the same representation as for the key exchange, that is, given $([\mathfrak{l}_1], \ldots, [\mathfrak{l}_u])$, express any element $[\mathfrak{g}_1]^{e_1} \ldots [\mathfrak{g}_g]^{e_g}$ 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 $([\mathfrak{l}_1], \ldots, [\mathfrak{l}_u])$. 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 2.

Algorithm 1 Key Recovery

Input : The elements $([\mathfrak{l}_1], \ldots, [\mathfrak{l}_u])$, two curves E_0 and E_A defined over $\mathbb{F}_p$, a generating set of $\mathcal{C}\ell(\mathcal{O})$: $([\mathfrak{g}_1], \ldots, [\mathfrak{g}_g])$
Output : A vector $(e_1, \ldots, e_u)$ 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 3.
2: Apply Kuperberg's Algorithm on f and g, get $[\mathfrak{s}]$.
3: Using Algorithm 3, decompose $[\mathfrak{s}]$ as $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$ with small e_i.
4: **return** $(e_1, \ldots, e_u)$

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 $([\mathfrak{l}_1], \ldots, [\mathfrak{l}_u])$ and some curve E. This cost will be studied in the next sections.

4.2 Hidden shift algorithm for cyclic groups

In this section, we present a method to use a hidden shift algorithm in $\mathbb{Z}/(2^n)\mathbb{Z}$ to solve a hidden shift problem in $\mathbb{Z}/N\mathbb{Z}$, in order to have the concrete estimates we need. The idea is to see the hidden shift in $\mathbb{Z}/N\mathbb{Z}$ as a hidden subgroup problem

in $\mathbb{Z}/2\mathbb{Z} \ltimes \mathbb{Z}$ (the subgroup being $\langle (0,N),(1,s)\rangle$, and to solve this subgroup problem in $\mathbb{Z}/2\mathbb{Z} \ltimes \mathbb{Z}/2^n\mathbb{Z}$, for a suitable n.

Kuperberg's algorithm uses an oracle that must produce elements of the form $\frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2i\pi\frac{xs}{2^n}\right)|1\rangle\right)$, that we note $|\psi_x\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \chi(\frac{xs}{2^n})|1\rangle\right)$, for a random, classically known x. Such an oracle is easy to implement with two functions that satisfy $f(x) = g(x+s)$ in $\mathbb{Z}/2^n\mathbb{Z}$, using two quantum Fourier transforms and some measurements.

If we manage to create an oracle that will produce qubits close enough to the ones needed, then the algorithm will perform correctly with a good enough probability.

First, we sample on all the possible values and compute the image through f and g, that is, we have the state

$$\frac{1}{2^{(n+1)/2}}\sum_{x=0}^{2^n}|0\rangle|x\rangle|f(x \mod N)\rangle + |1\rangle|x\rangle|g(x \mod N)\rangle \ .$$

Then, we measure the output register, which projects the input registers to

$$\frac{1}{\sqrt{X}}\left(|0\rangle|x_1\rangle + |1\rangle|x_2\rangle + \sum_{i=0}^{\lfloor 2^n/N\rfloor}|0\rangle|x+iN\rangle + |1\rangle|x+iN+s\rangle\right) \ .$$

The first two elements are the problematic ones, as they do not satisfy the periodicity constraint. They may not always be there, but we consider the worst situation. Now, if we apply a Quantum Fourier Transform on the second register, and we obtain

$$\frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{X}}\left(\chi\left(\frac{x_1y}{2^n}\right)|0\rangle + \chi\left(\frac{x_2y}{2^n}\right)|1\rangle + \chi\left(\frac{xy}{2^n}\right)\left(\sum_{i=0}^{\lfloor 2^n/N\rfloor}\chi\left(\frac{iNy}{2^n}\right)\right)\left(|0\rangle + \chi\left(\frac{sy}{2^n}\right)|1\rangle\right)\right)|y\rangle \ .$$

This state is a noisy version of the wanted state, and we need this noise to be small enough not to disrupt the computation.

The component $|0\rangle + \chi\left(\frac{sy}{2^n}\right)|1\rangle$ has an amplitude of

$$\frac{1}{\sqrt{Y}}\left|\frac{1-\chi\left(\frac{\lceil 2^m/N\rceil Ny}{2^n}\right)}{1-\chi\left(\frac{Ny}{2^n}\right)}\right| \ .$$

Using arguments *à la* Shor, we find that this amplitude is greater than $\frac{2}{\sqrt{N}\pi}$ if $|Ny| \mod 2^n \leq N/2$. We have N values in that fulfils this property. Hence, we will measure elements with a small enough noise with probability greater than $4/\pi^2$. Moreover, y is classically known, which means we can drop any qubit that does not fit in this range. If we need Q queries, we need to have a noise probability in $1/Q$ to have a fixed success probability. As the noise is in $\frac{N^2}{2^{2n}}$, we can take $2^n \simeq N\sqrt{Q}$. Here, we will consider $n = \left\lceil \log_2(N) + \sqrt{\log_2(3)/2\log_2(N)}\right\rceil$. The success

probability will be greater than $1/e$, when we add an additional cost of e to the algorithm. For Kuperberg's algorithm, we will not sample the elements uniformly in $\{0, \ldots, 2^n - 1\}$, but we will sample in the interval $N^{-1}\{-N/2, \ldots, N/2\}$. This is a very biased sampling, but after $\sqrt{2\log_2(3)n}$ collisions, we can reach any element, hence the algorithm will behave as expected.

Table 2. Cost estimates for the hidden shift algorithm.

$\lvert\log_2(p)\rvert$	n	Hidden shift cost ($\log_2$)	Memory cost ($\log_2$)
512	271	32	29.5
1024	533	44	41
1792	923	57	54

4.3 Finding a Basis of Multi-periods

The ideas in this section and the next one are fairly close to the ones in [2], 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 [2]). 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 [23], 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 $[\mathfrak{g}_1]^{j_1} \dots [\mathfrak{g}_k]^{j_k} = \mathbf{1}$ implies $j_1 = 0 \mod ord([\mathfrak{g}_1])$, $\dots j_k = 0 \mod ord([\mathfrak{g}_k])$, 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 $\mathcal{L}$. In general, $\mathcal{L}$ may have a non full rank. We upper bound its rank by u; if it turns out to be lower, the complexity of the lattice algorithms will be improved.

4.4 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) [22]. 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 [15] and [6].

We do not study quantum lattice algorithms (see e.g [20]), since approximation algorithms are sufficient for us.

Remark 3 (Basis Quality Benchmarks). In works such as [6], 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:

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

From [15], 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 [15], 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 $||b_1||_2$. Since the whole class group $\mathcal{C}\ell(\mathcal{O})$ is spanned by products of powers of the $[\mathfrak{l}_i]$ between $-m$ and m, we have $\lambda_1(\mathcal{L}) \leq 2m\sqrt{u}$, hence $||b_1||_2 \leq 2c^u m\sqrt{u}$.

Algorithm 2 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 $([\mathfrak{g}_1], \ldots, [\mathfrak{g}_g])$

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

Remark 4. There exists a classical subexponential algorithm to compute the class group [17]: Algorithm 2 could be performed classically in subexponential time (this is done by Couveignes in [9]). However, since we consider a quantum adversary, it makes sense to use quantum polynomial-time algorithms to compute the class group structure.

4.5 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 2. 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 [13], 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 [15] and [6] 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 $\{-m, \ldots, m\}$.

We have:

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

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

Algorithm 3 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 2 with Hermite factor c.

Output : A vector $\bar{s}$ such that $||s||_1 \leq u^{3/2} mc^u$ 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 $||\bar{t} - \bar{v}||_1 \leq u^{3/2} mc^u$.
2: **return** $\bar{t} - \bar{v}$.

Algorithm 3 and Algorithm 2 above are the two main components of Algorithm 7 in [2]. The authors also use BKZ to reduce the lattice base and Babai's algorithm to solve the approximate CVP instance. They however considered 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.

We are now able to give the complexity for all three NIST levels given in [5] (Table 3). 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, counted in equivalent classical group actions, our final complexity estimates remain apparently small.

Table 3. Summary of the cost overhead on the computation of the group action, w.r.t. a legimitate key exchange computation.

Level	$\lvert\log_2 p\rvert$	u	m	Overhead
NIST 1	512	74	5	$22 \simeq 2^5$
NIST 3	1024	128	8	$57 \simeq 2^6$
NIST 5	1792	200	11	$180 \simeq 2^8$

Remark 5. Once the secret isogeny has been recovered, the factors in Table 3 also give the multiplicative overhead on the size of the representation found w.r.t the original one.

5 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 [9], 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 $\{-m, \ldots, m\}$ 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 $\mathcal{C}\ell(\mathcal{O})$: $[\mathfrak{l}_1], \ldots, [\mathfrak{l}_u]$ 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 should 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 [15] 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.

In order to have a garanteed efficient asymptotical algorithm, one should apply in this case the general method given in [2] (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 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 [12]. In [12], 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.[4] 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:

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

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).

[4] 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.

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:

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

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 4.4) 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 [12], 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 $4(u+v)c^{u+v} \simeq 416 \leq 2^9$.

Improving this analysis would require to bound the multiplicative speedup not by $\max \left|\frac{e_i}{m_i}\right|$, but with more precision (using the respective computation costs of all isogenies considered).

If we computed instead the class group action using a maximal basis of Elkies primes, dismissing the refined structure from [12], we would obtain a dimension 162 in this example (with a 511-bit prime). The multiplicative overhead given by lattice reduction would be $98 \leq 2^7$, 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^{41} time and $2^{29.5}$ memory. Recall that the cost of the group action is overestimated here. 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.

6 Conclusion

The parameters set in [5] seem to provide only around half of the expected security, as summarized in Table 4.

Table 4. Summary of the attack cost versus the expected cost, in $\log_2$ scale.

Level	Memory	Queries	Cost per query	**Total Cost**	Expected Cost
NIST 1	29.5	32	5	**37**	62
NIST 3	41	44	6	**50**	94
NIST 5	54	57	8	**65**	129

Protecting CSIDH Against this Attack. Multiplying the base field bit size by 4 seems to increase the cost of this attack to meet the expected quantum security. However, one would need to find a suitable p, and this would also multiply the cost of the key exchange and the size of the keys by a similar factor.

Benchmark. Since the classical complexity estimates are given in equivalent group action computations, we do the same for quantum complexity estimates. This of course induces an overhead on the precise *gate count* of a quantum attacker. As further justification, notice that the NIST levels of security refer to exhaustive key search on AES. Precise quantum resource estimates of Grover's algorithm applied to AES [16] have shown that the implementation of a reversible AES adds such an overhead (approximately 2^{20} gates) to the respective 2^{64}, 2^{96} and 2^{128} complexities.

Other Isogeny-based Schemes. The NIST candidate SIKE [11] does not seem to be affected by this attack, as it uses supersingular elliptic curves on $\mathbb{F}_{p^2}$.

We showed 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 [12, 5, 2], by setting up a quantum class group action oracle alternative to the one given in [8], and we provided explicit cost estimates.

Very recently, De Feo, Kieffer and Smith [12] 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. Due to a reduction in dimension (less ideals are used to span the whole class group), the quantum speedup could even be better than the classical one.

We conclude that, when evaluating the security of an isogeny-based primitive using the commutative action of $\mathcal{C}\ell(\mathcal{O})$, such as CRS or CSIDH, the quantum superposition oracle for computing the action of $\mathcal{C}\ell(\mathcal{O})$ should be accounted for *a certain amount* of classical evaluations of this action. This number depends on how efficiently the lattice reduction steps above can be performed, but it can be made small. 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 and Jean-François Biasse for pointing out the reference [2].

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. 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)
3. 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)
4. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. Cryptology ePrint Archive, Report 2018/432 (2018), https://eprint.iacr.org/2018/432
5. 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
6. 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)
7. 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
8. 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
9. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
10. 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
11. 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
12. 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
13. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press (2012), https://www.math.auckland.ac.nz/ sgal018/crypto-book/crypto-book.html
14. 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
15. 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)

16. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's Algorithm to AES: Quantum Resource Estimates. In: PQCrypto. Lecture Notes in Computer Science, vol. 9606, pp. 29–43. Springer (2016)
17. 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)
18. 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
19. 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
20. 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
21. Regev, O.: A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space. CoRR (2004), http://arxiv.org/abs/quant-ph/0406151
22. 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
23. 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