Submerging CSIDH

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 for post-quantum non-interactive key-exchange, presented at ASIACRYPT 2018. Based on supersingular elliptic curve isogenies, it is similar in design to a previous scheme by Couveignes, Rostovtsev and Stolbunov, but aims at an improved balance between efficiency and security. In the proposal, the authors suggest concrete parameters in order to meet some desired levels of quantum security. These parameters are based on the hardness of recovering a hidden isogeny between two elliptic curves, using a quantum subexponential algorithm of Childs, Jao and Soukharev. This algorithm combines two building blocks: first, a quantum algorithm for recovering a hidden shift in a commutative group. Second, a computation in superposition of all isogenies originating from a given curve, which the algorithm calls as a black box.

In this paper, we give a comprehensive security analysis of CSIDH. Our first step is to revisit three quantum algorithms for the abelian hidden shift problem from the perspective of non-asymptotic cost. There are many possible tradeoffs between the quantum and classical complexities of these algorithms and all of them should be taken into account by security levels. Second, we complete the non-asymptotic study of the black box in the hidden shift algorithm.

This allows us to show that the parameters proposed by the authors of CSIDH do not meet their expected quantum security.

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

1 Introduction

Problems such as factoring and solving discrete logarithms, believed to be classically intractable, underlie the security of most asymmetric cryptographic primitives in use today. After Shor found a quantum polynomial-time algorithm for both [36], the cryptographic community has been actively working on replacements, culminating with the ongoing NIST call for post-quantum primitives [31].

One of the families of problems studied concerns elliptic curve isogenies. In this setting, we consider a graph, whose vertices are elliptic curves, and whose edges are non constant morphisms (isogenies). The main hard problem underlying the security of these new systems, that should remain hard for a quantum attacker, is finding a path between two curves. Originally, a key-exchange based on ordinary curves was proposed independently by Rostovtsev and Stolbunov [37] and Couveignes [14]. Later, a quantum algorithm was given in [12], that could find an isogeny between two such curves in subexponential time, a problem for which classical algorithms still require exponential time. Although it is not broken in quantum polynomial time, the scheme became considered as too inefficient with respect to its post-quantum security.

Meanwhile, cryptography based on *supersingular* elliptic curves isogenies was proposed [15]. In the NIST standardization process, the candidate SIKE, which is a supersingular isogeny key-exchange protocol, was selected for the second round. Indeed, the quantum algorithm for finding ordinary isogenies cannot be applied for supersingular curves, whose graphs have a different structure. The best known quantum algorithm for breaking SIKE has an exponential time complexity. Later works on the security of the system have shown that it may be even more secure than claimed, classically [1] and quantumly [25], and its original key sizes may be decreased for more efficiency.

CSIDH. CSIDH is a new primitive proposed in [9]. Its name stands for "commutative supersingular isogeny Diffie-Hellman", since its point is to make isogeny-based key exchange efficient in the *commutative* case. CSIDH uses supersingular elliptic curves, with the additional constraint that they have to be defined over $\mathbb{F}_p$. In this case, the $\mathbb{F}_p$-isogeny graph has a structure analogous to the ordinary isogeny graph, and the subexponential quantum algorithm of [12] can still be used to recover an isogeny between two given curves.

The difference with the Couveignes–Rostovtsev–Stolbunov scheme is that CSIDH is defined in a more favorable setting: the isogenies can be computed faster. Having a quantum subexponential attack on the scheme is not a fatal issue, if the balance between efficiency and security can be turned to its advantage. However, this puts CSIDH in a peculiar situation. To the best of our knowledge, it is the only post-quantum scheme actively studied[3] against which a quantum adversary enjoys more than a polynomial speedup. Indeed, all other schemes, whether they are based on lattices, codes, or supersingular isogenies, rely on problems whose structure seems harder to exploit by a quantum adversary. The quantum acceleration seems at best quadratic, and even this speedup can be difficult to achieve. For example, the asymptotic time for breaking SIKE goes from a classical T to a quantum $T^{2/3}$ only.

In only one year, CSIDH has been the subject of many publications, showing a renewed interest of the community for the protocols based on commutative elliptic curve isogenies. The similarity of CSIDH to a regular non-interactive Diffie-Hellman key exchange, and its conjectured quantum resistance, have made it a very promising candidate for designing more asymmetric primitives based on isogenies. After its publication in [9], it has been used in [19] to devise the

[3] Unfortunately, CSIDH was published at ASIACRYPT 2018, hence after the beginning of the NIST call, and it could not be submitted to the standardization process. The number of related publications clearly indicates that despite its outsider status, the scheme has attracted a wide interest.

signature scheme SeaSign. Systems based on elliptic curve isogenies tend to have small key sizes, but their computational efficiency is the main concern. CSIDH and SeaSign were further studied and improved in [29, 24, 28, 17], the last two works published at PQCRYPTO 2019.

Related work. Meanwhile, there has been a few works dealing with the security of CSIDH. As mentioned above, this security boils down to the *quantum* time for solving the underlying isogeny problem – the classical time will be massively higher. While most NIST candidates could simply assume at most a square root quantum speedup or take into account the asymptotic cost of the corresponding algorithm, the bounds of security for CSIDH are deemed to be much tighter. The asymptotic cost of running the CSIDH key-exchange was studied in [6], as it is a crucial factor of the quantum time complexity for breaking CSIDH. Furthermore, the number of quantum operations required to implement the 512-bit instance of the CSIDH key-exchange has been studied in full detail in [4], published at EUROCRYPT 2019. The authors remarked that a quantum adversary running the subexponential attack on the scheme would need already strong resources to implement it. This does not say however if the suggested CSIDH parameters indeed match the security levels conjectured in [9], and therefore, the computational efficiency of the scheme with respect to its security remained unclear.

In a *very* recent and independent work, Peikert [32] proposed some concrete cost estimates based on Kuperberg's second algorithm [27], that we also analyze in Section 3.4. As a different and complementary approach to ours, he allows large amounts of quantumly-adressable classical RAM, while we considered this algorithm with a small quantum memory and a large classical memory. The technicalities to retrieve the secret key are also different, and he proposed some simulations. The end result (the required number of group actions to compute) obtained is 2^{16}, for CSIDH-512, and a larger estimate for the bigger instances. Despite the differences, it is comparable with what we obtained. This is not surprising, as the core algorithm is the same, and the considered instances of the problem are rather small.

Contributions.

In this paper, we make a decisive move towards understanding the quantum security of CSIDH. We revisit three quantum abelian hidden shift algorithms from the available literature, that can be used to recover the secret key in a CSIDH key-exchange, from the point of view of non-asymptotic cost. We give a wide range of trade-offs between their quantum and classical time and memory complexities. This enables us to perform the first non-asymptotic quantum security analysis of CSIDH. We show that the parameters given in [9] do not meet the conjectured quantum security levels. Notably, the 512-bit parameters proposed can be attacked with a total of 2^{70} quantum gates (instead of an expected $2^{85.9}$), with 2^{85} classical computations. In isogeny-based cryptography, this is the first example of a "hybrid" attack that performs more classical than quantum computations. In order to assert security levels, we would recommend more conservative parameter choices, taking into account all known tradeoffs. Depending on the cost metric, the initial parameters may require to be multiplied by a factor roughly between 6 and 12, which would have a significant impact on the efficiency of the scheme.

Our study focuses on CSIDH, since this is currently the most studied cryptographic scheme with an attack based on abelian hidden shifts, but our results extend naturally to the original Couveignes–Rostovtsev–Stolbunov scheme or the improved version of [20].

Paper Outline.

The cost of an attack on CSIDH depends on two factors: first, the complexity (in queries, quantum and classical time) of an abelian hidden shift algorithm, which makes black-box quantum queries to a commutative group action. Second, the cost of a single query. The first part requires knowledge of quantum computing, but not of the inner workings of CSIDH. The second part requires to know which group we are dealing with and how many operations an isogeny computation represents.

Section 2 below presents some preliminaries of quantum computing, the context of the CSIDH group action, and the outline of the attack on CSIDH. We next go into the details of this attack. In Section 3, we present the three main quantum algorithms for finding abelian hidden shifts. Our contribution here is to give non-asymptotic estimates of them, and to write a simple algorithm for cyclic hidden shift (Algorithm 2), which can be easily simulated. These simulations are available as Supplementary Material. In Section 4, we compute the cost of a query, putting together the cost of a quantum evaluation of the scheme and some lattice reduction overhead. We summarize our complexity analysis in Section 5 and show why the parameters of [9] do not meet the conjectured security levels.

2 Preliminaries

In this section, we first introduce some preliminaries of quantum computing and post-quantum security. Next, we present the CSIDH rationale. We close this section with the outline of the attack on CSIDH.

2.1 Quantum Computing

For the interested reader [30] provides a comprehensive introduction to quantum computing. We merely review some basic notions.

Since the early years of quantum computing, the *quantum circuit model* has arisen as the standard way of describing the computations running on a universal quantum computer. In this model, we stand at the logical level of quantum computing, not the implementation level. We do not estimate the physical resources required to emulate a quantum circuit.

A quantum circuit is made of wires and gates. Contrary to classical boolean circuits, whose wires and logical gates can be implemented as hardware, a quantum circuit represents a *sequence of operations* performed on some quantum system. Each wire represents a qubit, which is a two-dimensional quantum system. Qubits are often said to be the analogue of classical logical bits. The state of a qubit is represented by a vector in a two-dimensional Hilbert space $\mathcal{H}$, on an arbitrary basis $|0\rangle, |1\rangle$ which is denoted as the *computational basis.* In general, it is of the form $\alpha|0\rangle + \beta|1\rangle$ where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. The numbers $|\alpha|^2$ and $|\beta|^2 = 1$ represent respectively the probabilities, upon measurement of the qubit, to obtain $|0\rangle$ or $|1\rangle$.

Due to the property of *entanglement*, the state of an n-qubit system lies in $\mathcal{H}^{2^n} = \mathcal{H}^{\otimes n}$, the *tensor product* of n copies of $\mathcal{H}$. It has 2^n basis vectors, which are all n-bit strings. Measuring gives one of this n-bit strings with some probability corresponding to its amplitude. In practice, considering the tensor of two quantum states simply means that we put them together.

A quantum computation starts with all qubits initialized to $|0\rangle$ and disentangled. It then performs a sequence of self-adjoint (linear) operators of $\mathcal{H}^{2^n}$ and measurements. During the computation, due to constructive *and destructive* interferences, the probability amplitudes of the states on the computational basis are modified. All amplitude should be moved towards n-big strings giving a meaningful result (for example, towards the expected secret).

An ubiquitous building block in the quantum algorithms studied in this paper is the Quantum Fourier Transform (QFT). We will note $\chi(x) = \exp(2i\pi x)$. The QFT maps a state $|k\rangle$, representing a number between 0 and $N-1$, to:

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \chi\left(\frac{jk}{N}\right) |j\rangle \quad .$$

Its name stems from the analogy with a classical discrete Fourier transform, although the result is a superposition. If the input is $|0\rangle$ (0 modulo N), we get a uniform superposition over all $j \in \{0, \dots, N-1\}$: $\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$.

Except in the case where we assume an *online oracle access* to some quantum black-box, which is not the case in this paper, the computations performed can be ultimately broken down into *quantum gates.* If the qubits are analog of classical bits, then quantum gates are analog of classical gates. They represent a set of basic operations with cost 1. Most of the times, a single processor model is assumed. At each time step, the quantum computer applies a gate to a qubit, or to a pair of qubits; so it makes sense to define the quantum time complexity as the total number of gates. The term "ancilla qubits" refers to additional qubits initialized to $|0\rangle$ and returned afterwards to this state. The less ancillas we use, the more often they have to be returned to $|0\rangle$. This can induce an overhead in quantum gates.

There exists multiple *universal gate sets*, all of which allow to perform all meaningful quantum computations. We can only compare quantum time complexities between two circuits written with the same gate set. Since we wish to compare some of our complexities with [23], we use the standard "Clifford+T" set for all our benchmarks. The Clifford set contains the controlled-Z (or CNOT) gate and the Hadamard gate. The (single-qubit) Hadamard gate maps $|0\rangle$ to $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. The (two-qubit) CNOT gate maps $|x\rangle|b\rangle$ to $|x\rangle|x \oplus b\rangle$. The (single-qubit) T-gate maps $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $\exp i\pi/4\,|1\rangle$. In currently envisioned quantum computing architectures, the T-gate seems the most costly to implement fault-tolerantly, due to its behavior with quantum error correcting codes. It is possible to realize the Toffoli gate with 7 T-gates.

2.2 What is Post-Quantum Security?

Despite the massive investments in engineering and research, quantum computers for "practical" cryptographic applications, such as finding RSA or ECDSA keys, do not yet exist. However, post-quantum cryptography is a blooming field. It relies on three assertions of common sense: first, updating standards is a slow process. Second, if a quantum adversary appears at any point in the future, all secrets shared with quantum-unsafe cryptography will be leaked. Third, parameters should be designed with a strong margin with respect to the best attacks achieved in practice. In our case, we should not wait until a quantum computer actually breaks some RSA parameters (if it does indeed).

With this in mind, the study of post-quantum schemes becomes relevant, even if they have higher key sizes or computational cost than before. However, as we are studying security with respect to a non-existent computer, we have to agree on a common ground, in order to make security assumptions and claims similarly to the classical setting. To date, such common ground is provided by the NIST call [31]. While it is impossible to assert the real time, in seconds, of running a quantum computation, we have the circuit model, in which we can count operations. We do not know which universal gate set will be used or what overhead we might expect for emulating a quantum circuit. We overcome this by making comparisons instead.

The NIST provided 5 security levels, among which levels 1, 3 and 5 are considered in [9]. These levels correspond respectively to a key-recovery on AES-128, on AES-192 and AES-256. A cryptographic scheme, instantiated with some parameter size, matches level 1 if there is no quantum key-recovery running faster than quantum exhaustive search of the key for AES-128, and classical key-recovery running faster than classical exhaustive search. The corresponding quantum gate cost for the Clifford+T set ($2^{85.9}$, using Grover search) is given in [23]. This means also that an algorithm making less than $2^{85.9}$ quantum computations (counted in the Clifford+T set) and 2^{128} classical computations breaks the NIST level 1 security, as it runs below the security level of AES. Hence, we can say that level 1 corresponds to $2^{85.9}$ Clifford + T gates and 2^{128} classical computations.

There can be some refinements of the cost metrics. For example, is the comparison sound if an algorithm uses a lot of memory, and the other one much less? This is a meaningful question classically, and also quantumly, as maintaining the coherence of a massive amount of qubits should be much harder than for a smaller set. The NIST call also gives an additional parameter MAXDEPTH. It supposes that the *depth* of the attacker's quantum circuit may be limited to, say, 2^{40}. The depth is the length of the longest sequence of gates in the circuit. There exists quantum algorithms inherently easy to parallelize, in which many gates can be applied concurrently. We do not consider MAXDEPTH in this paper, and merely point out that most of the algorithms we present can run efficiently with limited depth.

Simulations in the Quantum Circuit Model. Some of our work relies on simulations of Kuperberg's algorithm for finding an abelian hidden shift. This algorithm can be described as a single, massive quantum circuit, but it is better understood as a sequence of quantum computations and measurements. The result of each measurement controls the next quantum computations applied. The precise time complexity, *i.e.* the number of quantum gates in total, can be analyzed asymptotically. But since intermediate measurements are probabilistic, the most precise way to study the time complexity of this algorithm is to simulate it, by randomly selecting outputs at each measurement and counting the number of subsequent operations to perform. The key fact is that these simulations remain sound in the quantum circuit model. They give an *average quantum time* which is meaningful, although not exact.

2.3 Context of CSIDH

Let $p > 3$ 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 curves *defined over* $\mathbb{F}_p$ is special. When $\mathcal{O}$ is an order in an imaginary quadratic field, each supersingular elliptic curve defined over $\mathbb{F}_p$ having $\mathcal{O}$ as its $\mathbb{F}_p$-rational endomorphism ring corresponds to an element of $\mathcal{C}\ell(\mathcal{O})$, the ideal class group of $\mathcal{O}$. 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 $\mathcal{O}$.

One-way Group Action. All use cases of the CSIDH scheme can be pinned down to the definition of a *one-way group action* (this is also the definition of a *hard homogeneous space* by Couveignes [14]). A group G acts on a set X. Operations in G, and the action $g * x$ for $g \in G, x \in X$, are easy to compute. Recovering g given x and $x' = g * x$ is hard. In the case of CSIDH, X is a set of Montgomery curves of the form $E_A : y^2 = x^3 + Ax^2 + x$ for $A \in \mathbb{F}_p$, 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.

This action of the class group already exists in the ordinary case, which is the reason of the similarity between CSIDH and the Couveignes – Rostovtsev

– Stolbunov scheme. Quantum algorithms for recovering abelian hidden shifts solve exactly this problem of finding g when G is commutative. There exists a family of such algorithms, initiated by Kuperberg. The variant of [12] targets precisely the context of ordinary curves, and it can be applied to CSIDH. But a comprehensive security analysis of the scheme should consider all algorithms available in the literature.

Representation of $\mathcal{C}\ell(\mathcal{O})$. The prime p is chosen with a very specific form: $p = 4 \cdot \ell_1 \cdots \ell_u - 1$ 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.

This enables to represent the elements of $\mathcal{C}\ell(\mathcal{O})$ (hence, the isogenies) in a way that is now specific to CSIDH, and the main reason of its efficiency. 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 $\mathcal{O}$. The image curves by these ideals can be computed efficiently [9, Section 8].

The authors of CSIDH consider a subset of $\mathcal{C}\ell(\mathcal{O})$, with all ideals of the form $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$ where $e_i \in \{-m \ldots m\}$ for some small m, and $[\mathfrak{l}_i]$ is the class of $\mathfrak{l}_i$. If we suppose that these products fall randomly in $\mathcal{C}\ell(\mathcal{O})$, which has $O(\sqrt{p})$ elements, it suffices to take $2m + 1 \simeq p^{1/(2u)}$ in order to span the group $\mathcal{C}\ell(\mathcal{O})$ or almost all of it. Since a greater m yields more isogeny computations, u should be the greatest possible. With this constraint in mind, we derive optimal parameters for the three security levels from [9] in Table 1.

Table 1: Approximate parameters for the three security levels of [9].

Level	Expected quantum security	$\log_2 p$	u	m
NIST 1	As hard as AES-128 key-recovery	512	74	5
NIST 3	As hard as AES-192 key-recovery	1024	132*	7*
NIST 5	As hard as AES-256 key-recovery	1792	209*	10*

*In [9], a prime p is given only for the first instance. The value of u given for the other instances is an upper bound.

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 $\bar{e} = (e_1 \ldots e_u)$. Computing operations in the class group is easy and only costs multiplications and inversions modulo p. Given an element of $\mathcal{C}\ell(\mathcal{O})$ of the form $[\mathfrak{b}] = \prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$, we compute $E' = [\mathfrak{b}] \cdot E$ by applying a sequence of $\sum_i e_i$ isogenies. The CSIDH public keys are curves. The secret keys are isogenies with this representation.

CSIDH Original Security Analysis in [9]. The problem underlying the security of CSIDH is: given an entry point E_0 in the isogeny graph, given a Montgomery curve E_A, recover 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.

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, they use the cost given in [12] for ordinary curves:

$$\exp\left((\sqrt{2}+o(1))\sqrt{\log N \log\log N}\right), \quad \text{where } N = \#\mathcal{C}\ell(\mathcal{O})$$

2.4 Attack Outline

Given E_A, we 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 L_1 norm 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 little more than for the legitimate user. All our concrete estimates will be given in the next sections.

An attack on CSIDH runs as Algorithm 1.

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 $f : [x] \in \mathcal{C}\ell(\mathcal{O}) \mapsto [x] \cdot E_0$ and $g : [x] \in \mathcal{C}\ell(\mathcal{O}) \mapsto [x] \cdot E_A$.
2: Apply a quantum abelian hidden shift algorithm, which recovers the "shift" between $f$ and $g$ using a certain number of queries to them. Obtain $[\mathfrak{s}]$
3: Decompose $[\mathfrak{s}]$ as $\prod_{i=1}^{u} [\mathfrak{l}_i]^{e_i}$ with small $e_i$.
4: return $(e_1, \dots, e_u)$
```

To evaluate these functions f and g, we need to compute $[x] \cdot E$ for $[x]$ in $\mathcal{C}\ell(\mathcal{O})$, *in superposition over* $[x]$, for some curve E. This means, in short, computing $[x] \cdot E$ for any element of the class group. In [12], in the context of ordinary curves, the authors show how to decompose any ideal as a product of smaller ones, and they succeed in bounding the cost of evaluating the corresponding isogenies by a subexponential factor.

The crucial difference with CSIDH is that the scheme provides an additional structure, which can be used to compute $[x] \cdot E$ more efficiently. Indeed, we have supposed that the class group is spanned by products of the form $[\mathfrak{l}_1]^{y_1} \dots [\mathfrak{l}_u]^{y_u}$ with small y_i. If we are able to rewrite $[x]$ as such a product, then the evaluation of $[x] \cdot E$ costs no more than the "legitimate" evaluation of the scheme. This is where lattice reduction intervenes. It was mentioned in [14], considered in [5] to obtain short representations of large-degree isogenies as elements of $\mathcal{C}\ell(\mathcal{O})$ and further followed in [6] where, independently from our work, the authors study how to attack the CSIDH scheme using this efficient isogeny evaluation oracle.

The (quantum) cost of the attack is roughly:

$$\begin{array}{c}\text{Number of evaluations of}\\ f \text{ and } g\end{array} \times \begin{array}{c}\text{Cost of computing } [x]\cdot E \text{ for}\\ \text{any } [x] \text{ in superposition}\end{array}$$

and there are some classical pre- and postcomputations.

In general, the short representation of x is not as short as in a“legitimate” CSIDH key-exchange. There is some approximation overhead, related to the quality of the lattice reduction that we performed. Our analysis will show that this overhead is minor (less than 2^3) for the CSIDH original parameters, but it is actually asymptotically subexponential (see [6]). The total cost becomes:

$$\begin{array}{c}\text{Number of}\\ \text{evaluations of } f\\ \text{and } g\end{array} \times \begin{array}{c}\text{Lattice reduction}\\ \text{overhead}\end{array} \times \begin{array}{c}\text{Cost of computing } [x]\cdot E \text{ for an}\\ [x] \text{ of the form } [\mathfrak{l}_1]^{e_1}\dots[\mathfrak{l}_u]^{e_u},\\ \text{with } -m \le e_i \le m.\end{array}$$

Since we give for the first time non-asymptotic estimates for the first and second terms, we now can use the estimations of [4] for the third term, with two limitations. First, the authors of [4] gave an exact number of operations, but only for a CSIDH-512 isogeny evaluation; we need to extrapolate roughly for bigger parameters. Second, they consider the same bound m for all exponents. As some isogenies are easier to compute than others (for example $[\mathfrak{l}_1]$ is faster than $[\mathfrak{l}_u]$), one could take different bounds. This means that our quantum gate counts may be slightly overestimated.

3 Quantum Abelian Hidden Shift Algorithms

In this section, we recall the literature on quantum algorithms for solving the hidden shift problem in commutative (abelian) groups and present in detail three algorithms. For each of them, we give tradeoff formulas and some non-asymptotic estimates in the context of CSIDH. The first one (Algorithm 2) is a new variant of [26] for cyclic groups, whose behavior is easy to simulate. The second is by Regev [33] and Childs, Jao and Soukharev [12]. The third is Kuperberg's second algorithm [27].

3.1 Context

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 commutative groups, there exists quantum subexponential algorithms. The first result on this topic was an algorithm with low query complexity, by Ettinger and Høyer [18], 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 [26]. His 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 his 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 [33], which has been generalized by Kuperberg in [27], with an algorithm in $\widetilde{O}\left(2^{\sqrt{2\log_2(N)}}\right)$ quantum queries and classical memory, and a polynomial quantum memory. Regev's variant has been generalized to arbitrary commutative groups in the appendix of [12], with the same complexity. A complexity analysis of this algorithm with tighter exponents can be found in [7].

Cyclic Groups and Concrete Estimates. The $\widetilde{O}$ are not practical for concrete estimates of quantum costs. However, in [8], 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. In the context of CSIDH, the class group $\mathcal{C}\ell(\mathcal{O})$ does not have a power of 2 as cardinality, but in most cases, its odd part is cyclic, as shown by the Cohen–Lenstra heuristics [13]. So we choose to approximate the class group as a cyclic group. This is why we propose in what follows a generalization of [8, Algorithm 2] that works for any N, at essentially the same cost.

In order to apply Kuperberg's algorithm, we need a representation of the class group. We can use the quantum polynomial-time algorithm of [11], as done in [12]. 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 $([\mathfrak{g}_1], \ldots, [\mathfrak{g}_g])$.

3.2 A First Hidden Shift Algorithm

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. This presentation, and the following, uses the notations of Section 2.1. We suppose an access to the quantum oracle that maps $|x\rangle|0\rangle|0\rangle$ to $|x\rangle|0\rangle|f(x)\rangle$, and $|x\rangle|1\rangle|0\rangle$ to $|x\rangle|1\rangle|g(x)\rangle$.

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

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

Then, we apply the quantum oracle, and get

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

We then measure the final register. We obtain a value $y = f(x_0) = g(x_0 + s)$ for some random x_0. The two first registers *collapse* on the superposition that corresponds to this measured value:

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

Finally, we apply a Quantum Fourier Transform (QFT) on the first register and measure it, we obtain a label ℓ and the state

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

The meaningful information in $\left|\psi_\ell\right\rangle$ does not lie in the respective probability amplitudes of $\left|0\right\rangle$ and $\left|1\right\rangle$, but in the phase $\chi\left(s\frac{\ell}{N}\right)$, which depends on s and $\frac{\ell}{N}$, and contains information on s. We now apply a *combination routine* on pairs of labeled qubits $\left(\left|\psi_\ell\right\rangle, \ell\right)$ as follows.

Combination Step. If we have obtained two qubits $\left|\psi_{\ell_1}\right\rangle$ and $\left|\psi_{\ell_2}\right\rangle$ with their corresponding labels ℓ_1 and ℓ_2, we can write the (disentangled) joint state of $\left|\psi_{\ell_1}\right\rangle$ and $\left|\psi_{\ell_2}\right\rangle$ as:

$$\left|\psi_{\ell_1}\right\rangle \otimes \left|\psi_{\ell_2}\right\rangle = \frac{1}{2}\left(\left|00\right\rangle + \chi\left(s\frac{\ell_1}{N}\right)\left|10\right\rangle + \chi\left(s\frac{\ell_2}{N}\right)\left|01\right\rangle + \chi\left(s\frac{\ell_1+\ell_2}{N}\right)\left|11\right\rangle\right) \quad .$$

We apply a CNOT gate, which maps $\left|00\right\rangle$ to $\left|00\right\rangle$, $\left|01\right\rangle$ to $\left|01\right\rangle$, $\left|10\right\rangle$ to $\left|11\right\rangle$ and $\left|11\right\rangle$ to $\left|10\right\rangle$. We obtain the state:

$$\frac{1}{2}\left(\left|00\right\rangle + \chi\left(s\frac{\ell_2}{N}\right)\left|01\right\rangle + \chi\left(s\frac{\ell_1+\ell_2}{N}\right)\left|10\right\rangle + \chi\left(s\frac{\ell_1}{N}\right)\left|11\right\rangle\right) \quad .$$

We measure the second qubit. If we measure 0, the first qubit collapses to:

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + \chi\left(s\frac{\ell_1+\ell_2}{N}\right)\left|1\right\rangle\right) = \left|\psi_{\ell_1+\ell_2}\right\rangle$$

and if we measure 1, it collapses to:

$$\frac{1}{\sqrt{2}}\left(\chi\left(s\frac{\ell_2}{N}\right)\left|0\right\rangle + \chi\left(s\frac{\ell_1}{N}\right)\left|1\right\rangle\right) = \chi\left(s\frac{\ell_2}{N}\right)\left|\psi_{\ell_1-\ell_2}\right\rangle \quad .$$

A common phase factor has no incidence, so we can see that the combination either produces $\left|\psi_{\ell_1+\ell_2}\right\rangle$ or $\left|\psi_{\ell_1-\ell_2}\right\rangle$, with probability $\frac{1}{2}$. Furthermore, the measurement of the first qubit gives us which of the labels we have obtained. Although we cannot choose between the two cases, we can perform favorable combinations: we choose ℓ_1 and ℓ_2 such that $\ell_1 \pm \ell_2$ is a multiple of 2 with greater valuation than ℓ_1 and ℓ_2 themselves.

Goal of the Combinations. In order to retrieve s, we want to produce the qubits with label 2^i and apply a Quantum Fourier Transform. 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 $\theta \in \left[0; \frac{1}{2^{n+1}}\right]$, 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 $n > \log_2(N)$.

From 2^n to any N. If N is a power of two, all of this works immediately. But we want to apply this simple algorithm to any cyclic group, with any N. The idea is to *not* take into account the modulo N in the combination of labels. We only want combinations such that $\sum_k \pm\ell_k = 2^i$. At each combination step, we expect the 2-valuation of the output label to increase (we collide on the lowest significant bits), but its maximum size can also increase: $\ell_1 + \ell_2$ is bigger than ℓ_1 and ℓ_2. 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.

Intuitively, the behavior of this algorithm will be close to the one of [8], as we only have a slightly higher amplitude in the values, and a few more elements to produce. The number of oracle queries Q is exactly the number of labeled qubits that are used during the combination step. Empirically, we only need to put 3 elements at each step in R in order to have a good success probability. This algorithm is easily simulated, because we only need to reproduce the combination step, by generating at random the new labels obtained at each combination. We estimate the total number of queries to be around $12 \times 2^{1.8\sqrt{n}}$.

We give the cost estimates for Algorithm 2 in Table 3, for group sizes corresponding to the CSIDH parameters. We consider the number of labeled qubits which are stored in memory, and the total number of oracle queries required to construct them. A slight overhead in time stems from the probability of success of $\frac{4}{\pi^2}$; the procedure needs to be repeated at most 4 times. In CSIDH, each query has a high quantum time complexity. The number of CNOT quantum

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 $i \leq n$ 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 $\{(2^i, i) | 0 \leq i \leq n\} \subset R$ 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 $|P_i| \geq 2$ 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
```

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

$\log_2(N)$	$\log_2(Q)$	$1.8\sqrt{\log_2(N)} + 2.3$
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

gates applied during the combination step (roughly equal to the number of labeled qubits at the beginning) is negligible. Notice also that the production of the labeled qubits can be perfectly parallelized.

Table 3: Cost estimates for Algorithm 2.

$\log_2(p)$	n	Queries ($\log_2$)	Number of qubits ($\log_2$)
512	256	33	31
1024	512	45	43
1792	896	58	56

3.3 An Approach Based on Subset-sums

The previous algorithm is only a variant of the first subexponential algorithm by Kuperberg in [26]. We develop here on a later approach used by Regev [33] and Childs, Jao and Soukharev [12] for odd N. Multiple tradeoffs have been analyzed in [7].

Subset-sum Combination Routine. The main idea is to change the way the labeled qubits are combined. Instead of a single CNOT, one can consider the system formed by k qubits:

$$\bigotimes_{i\le k} |\psi_{\ell_i}\rangle = \sum_{j\in\{0,1\}^k} \chi\left(\frac{j\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j\rangle$$

and apply

$$|x\rangle|0\rangle \mapsto |x\rangle|\lfloor x\cdot(\ell_1,\dots,\ell_k)/B\rfloor\rangle$$

for a given B.

Measuring the second register yields a value $V = \lfloor x\cdot(\ell_1,\dots,\ell_k)/B\rfloor$, the state becoming

$$\sum_{\lfloor j\cdot(\ell_1,\dots,\ell_k)/B\rfloor = V} \chi\left(\frac{j\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j\rangle \quad .$$

In order to get a labeled qubit, one can simply project on any pair (j_1, j_2) with j_1 and j_2 among this superposition of j. This is easy to do as long as the j are classically known. They can be computed by solving the equation $\lfloor j\cdot(\ell_1,\dots,\ell_k)/B\rfloor = V$, which is an instance of the subset-sum problem.

This labeled qubit obtained is of the form:

$$\chi\left(\frac{j_1\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j_1\rangle + \chi\left(\frac{j_2\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j_2\rangle$$

which, up to a common phase factor, is:

$$|j_1\rangle + \chi\left(\frac{(j_2 - j_1)\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j_2\rangle \quad .$$

We observe that the new label in the phase, given by $(j_2 - j_1)\cdot(\ell_1,\dots,\ell_k)$, is less than B. If we map j_1 and j_2 respectively to 0 and 1, we obtain a labeled qubit $|\psi_\ell\rangle$ with $\ell < B$. Now we can iterate this routine in order to get smaller and smaller labels, until the label 1 is produced.

If N is odd, one reaches the other powers of 2 by multiplying all the initial values by 2^{-a} and then applying normally the algorithm.

Algorithm 3 Combination routine

```
    Input: $(|\psi_{\ell_1}\rangle,\dots,|\psi_{\ell_k}\rangle)$, $r$
    Output: $|\psi_{\ell'}\rangle$, $\ell' < B$
 1: Tensor $\bigotimes_i |\psi_{\ell_i}\rangle = \sum_{j\in\{0,1\}^k}\chi\left(\frac{j\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j\rangle$
 2: Add an ancilla register, apply $|x\rangle|0\rangle \mapsto |x\rangle|\lfloor x\cdot(\ell_1,\dots,\ell_k)/B\rfloor\rangle$
 3: Measure the ancilla register, leaving with
        $V$ and $\sum_{\lfloor j\cdot(\ell_1,\dots,\ell_k)/B\rfloor = V}\chi\left(\frac{j\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j\rangle$
 4: Compute the corresponding $j$
 5: Project to a pair $(j_1, j_2)$.
    The register is now $\chi\left(\frac{j_1\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j_1\rangle + \chi\left(\frac{j_2\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|j_2\rangle$
 6: Map $|j_1\rangle$ to $|0\rangle$, $|j_2\rangle$ to $|1\rangle$
 7: Return $|0\rangle + \chi\left(\frac{(j_2-j_1)\cdot(\ell_1,\dots,\ell_k)}{N}s\right)|1\rangle$
```

There are 2^k sums, and N/B possible values, hence we can expect to have $2^kB/N$ solutions. If we take $k \simeq \log_2(N/B)$, we can expect 2 solutions on average. In order to obtain a labeled qubit in the end, we need at least two solutions, and we need to successfully project to a pair (j_1, j_2) if there are more than two solutions.

The case where a single solution exists cannot happen more than half of the time, as there are twice many inputs as outputs. We consider the case where we have strictly more than one index j in the sum. If we have an even number of such indices, we don't have any issue: we simply divide the indices j into a set of pairs, project onto a pair, and map transform one of the remaining indexes to 0 and the other to 1. If we have an odd number of such indices, since it is greater or equal than 3, we single out a solitary element, and do the projections as in the even case. The probability to fall on this element is less than $\frac{1}{t} \le \frac{1}{3}$ if there are t solutions, hence the probability of success in this case is more than $\frac{2}{3}$.

Complete Algorithm. The previous combination routine can be used recursively to obtain the value we want.

Linear number of queries. Algorithm 3 can directly produce the label 1 if we choose $k = \lceil \log_2(N) \rceil$ and $B = 2$. In that case, we will either produce 1 or 0 with a uniform probability, as the input labels are uniformly distributed.

If the group has a component which is a small power of two, the previous routine can be used with $B = 1$ in order to force the odd cyclic component at zero, at which point the algorithms of [8] can be used, with a negligible overhead.

Overall, the routine can generate the label 1 using $\log_2(N)$ queries with probability one half. This also requires to solve one subset-sum instances, which can be done in only $\widetilde{O}\left(2^{0.291 \log_2(N)}\right)$ classical time and memory [3].

We need to obtain $\log_2(N)$ labels, and then we can apply the Quantum Fourier Transform as before, to recover s, with a success probability $\frac{4}{\pi^2}$. So we expect to reproduce this final step 3 times. The total number of queries will be $8 \log_2(N)^2$, with a classical time and memory cost in $\widetilde{O}\left(2^{0.291 \log_2(N)}\right)$.

We note that this variant is the most efficient in quantum resources, as we limit the quantum queries to a polynomial amount. The classical complexity remains exponential.But we replace the complexity of a collision search (with an exponent 0.5) by that of the subset-sum problem (an exponent of 0.291).

In the case $N \simeq 2^{256}$, by taking into account the success probability of the final Quantum Fourier Transform, we can estimate the cost to be roughly 2^{19} quantum queries and 2^{86} classical time and memory.

Time/query tradeoffs. There are many possible tradeoffs, as we can adjust the number of steps and their sizes.

For example, we can proceed in two steps: the first step will produce values smaller than $\sqrt{N}$, and the second will use them to produce the value 1.

The subset-sum part of each step, done classically, will cost $\widetilde{O}\left(2^{0.291 \log_2(N)/2}\right)$ time and memory, and it has to be repeated $\log(N)^2/4$ times per value.

Hence, the total cost in queries is in $O(\log(N)^3)$, with a classical time and and memory cost in $\widetilde{O}\left(2^{0.291 \log_2(N)/2}\right)$.

For $N \simeq 2^{256}$, we will use Algorithm 3 to obtain roughly 130 values that are smaller than 2^{128}, and then apply Algorithm 3 on them to obtain the value 1. We can estimate the cost to be roughly 2^{24} quantum queries, 2^{60} classical time and 2^{45} memory.

This method generalizes to any number of steps. If we want a subexponential classical time complexity, then the number of steps has to depend of N. Many tradeoffs are possible, depending on the resources of the quantum attacker. This is developed in [7].

3.4 Kuperberg's Second Algorithm

This section revisits the algorithm from [27] and builds upon tradeoffs developed in [7]. We remark that the objects we were using in the two previous algorithms were of the form

$$|\psi_\ell\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \chi\left(s\frac{\ell}{N}\right)|1\rangle\right) \quad .$$

This is a particular case of qubit registers of the form

$$\left|\psi_{(\ell_0,\ldots,\ell_{k-1})}\right\rangle = \frac{1}{\sqrt{k}}\sum_{0\le i\le k-1}\chi\left(s\frac{\ell_i}{N}\right)|i\rangle \quad .$$

These multi-labeled qubit registers become the new building blocks. They are not indexed by a value ℓ, but by a vector $(\ell_0,\ldots,\ell_{k-1})$. We can remark that if we consider the joint state (tensor) of j single-label qubits $|\psi_{\ell_i}\rangle$, we directly obtain a multi-labeled qubit register of this form:

$$\bigotimes_{0\le i\le j-1}|\psi_{\ell_i}\rangle = \left|\psi_{\left(\ell'_0,\ldots,\ell'_{2^j-1}\right)}\right\rangle, \quad \text{with } \ell'_k = k\cdot(\ell_0,\ldots,\ell_{j-1}).$$

Combination Routine. These multi-labeled qubits, as before, can be combined to obtain better ones, by computing and measuring a partial sum, as presented in Algorithm 4.

Algorithm 4 A general combination routine

```
   Input: $(\left|\psi_{(\ell_0,\ldots,\ell_{M-1})}\right\rangle, \left|\psi_{(\ell'_0,\ldots,\ell'_{M-1})}\right\rangle) : \forall i, \ell_i < 2^a, \ell'_i < 2^a$, $r$
   Output: $\left|\psi_{(v_0,\ldots,v_{M'-1})}\right\rangle : \forall i, v_i < 2^{a-r}$
1: Tensor $\left|\psi_{(\ell_0,\ldots,\ell_{M-1})}\right\rangle\left|\psi_{(\ell'_0,\ldots,\ell'_{M-1})}\right\rangle = \sum_{i,j=0}^{M-1}\chi\left(\frac{s(\ell_i+\ell'_j)}{N}\right)|i\rangle|j\rangle$
2: Add an ancilla register, apply $|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle\left|\lfloor(\ell_i+\ell'_j)/2^{a-r}\rfloor\right\rangle$
3: Measure the ancilla register, leaving with
       $$V \text{ and } \sum_{i,j:\lfloor(\ell_i+\ell'_j)/2^{a-r}\rfloor=V}\chi\left(\frac{s(\ell_i+\ell'_j)}{N}\right)|i\rangle|j\rangle$$
4: Compute the $M'$ corresponding pairs $(i,j)$
5: Apply to the state a transformation $f$ from $(i,j)$ to $[0;M'-1]$.
6: Return the state and the vector $v$ with $v_{f(i,j)} = \ell_i + \ell'_j$.
```

If Algorithm 3 was essentially a subset-sum routine, Algorithm 4 is a 2-list merging routine. Step 4 simply consists in iterating trough the sorted lists of $(\ell_0,\ldots,\ell_{M-1})$ and $(\ell'_0,\ldots,\ell'_{M-1})$ to find the matching values (and this is exactly a classical 2-list problem). Hence, it costs $\widetilde{O}(M)$ classical time, with the lists stored in classical memory. The memory cost is $\max(M,M')$. The quantum cost comes from the computation of the partial sum and from the relabeling. Both can be done sequentially, in $O(\max(M,M'))$ quantum time.

This routine can also be generalized to merge more than two lists. The only difference will be that at Step 4, we will need to apply another list-merging algorithm to find all the matching values. In particular, if we merge $4k$ lists, we can use the Schroeppel-Shamir algorithm [35], to obtain the solutions in $O(M^{2k})$ classical time and $O(M^k)$ classical memory.

Once we are finished, we have to project the vector to a pair of value whose difference is 1, as in Algorithm 3. Hence, the success probability is the same, better than 1/3.

Complete Algorithm. The complete algorithm uses the combination routine 4 recursively. As before, the final cost depends on the size of the lists, the number of steps and the number of lists we merge at each step. Then, we can see the algorithm as a merging tree.

The most time-efficient algorithms use 2-list merging. The merging tree is binary, the number of lists at each level is halved. We can save some time if we allow the lists to double in size after a merging step. In that case, the merging of two lists of size 2^m to one list of size 2^{m+1} allows to constrain $m-1$ bits[4], at a cost of $O(2^m)$ in classical and quantum time and classical memory. If we have e levels in the tree and begin with lists of size 2^{ℓ_0}, then the quantum query cost is $\ell_0 2^e$. The time cost will be in $\widetilde{O}\left(2^{\ell_0+e}\right)$, as the first step is performed 2^e times, the second 2^{e-1} times, and so on.

Allowing the lists to grow saves some time, but costs more memory. To save memory, we can also combine lists and force the output lists to be of roughly the same size. Hence, the optimal algorithm will double the list sizes in the first levels until the maximal memory is reached, when the list size has to stay fixed.

Overall, let us omit polynomial factors and denote the classical and quantum time as 2^t. We use at most 2^m memory and make 2^q quantum queries, begin with lists of size 2^{ℓ_0} and double the list sizes until we reach 2^m. Hence, the list size levels are distributed as in Figure 1. We have q equal the number of levels, and t equals the number of levels plus ℓ_0. As each level constrains as many bits as the log of its list size, the total amount of bits constrained by the algorithm corresponds to the hatched area.

Hence, with $\max(m,q) \leq t \leq m+q$, we can solve the hidden shift problem for $N < 2^n$ with

$$-\frac{1}{2}(t-m-q)^2 + mq = n$$

We directly obtain the cost of $\widetilde{O}\left(2^{\sqrt{2n}}\right)$ from [27] if we consider $t = m = q$.

Classical/Quantum Tradeoffs. The previous approach had the inconvenient of using equal classical and quantum times, up to polynomial factors. In practice, we can expect to be allowed more classical operations than quantum gates. We can obtain different tradeoffs by reusing the previous 2-list merging tree, and

[4] As in the end, we only need a list of size two, the bit we lose here is regained in the last step.

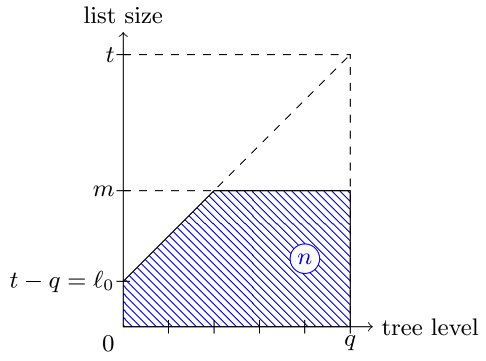

Fig. 1: Size of the lists in function of the tree level, in $\log_2$ scale, annotated with the different parameters.

seeing it as a 2^k-list merging tree. That is, we see k levels as one, and merge the 2^k lists at once. This allows to use the Schroeppel-Shamir algorithm for merging, with a classical time in $2^{2^k/2}$ and a classical memory in $2^{2^k/4}$. This operation is purely classical, as we are computing lists of labels, and it does not impact the quantum cost. Moreover, while we used to have a constraint on $\log(k)m$ bits, we now have a constraint on $(k-1)m$ bits.

For $k = 2$, omitting polynomial factors, with a classical time of 2^{2t} and quantum time of 2^t, a memory of 2^m, a number of quantum queries of 2^q and $\max(m, q) \leq t \leq m + q$, we can solve the hidden shift problem for $N < 2^n$ with

$$-\frac{1}{2}(t-m-q)^2 + mq = 2n/3$$

In particular, if we consider that $t = m = q$, we obtain an algorithm with a quantum time and query and classical memory complexity of $\widetilde{O}\left(2^{2\sqrt{\frac{n}{3}}}\right)$ and a classical time complexity of $\widetilde{O}\left(2^{4\sqrt{\frac{n}{3}}}\right)$.

Concrete estimates. If we consider $N \simeq 2^{256}$, then using the 2-list merging method we can succeed with 2^{23} initial lists of size 2, and doubling the size of the list at each level, until we obtain a list of size 2^{24}. In that case, we obtain classical and quantum time cost in 2^{39}, a classical memory in 2^{29} and 2^{34} quantum queries.

Using the 4-list merging, we can achieve the same in 10 steps with roughly 2^{55} classical time, 2^{23} classical memory, 2^{35} quantum time, 2^{31} quantum queries.

Other tradeoffs are also possible. We can reduce the number of queries by beginning with larger lists. We can also combine the k-list approach with the subset-sum approach to reduce the quantum time (or the classical memory, if we use a low-memory subset-sum algorithm).

For example, if we consider a 4-level tree, with a 4-list merging, an initial list size of 2^{24} and lists that quadruple in size, the first combination step can constrain $24\times 3-2=70$ bits, the second $26\times 3-2=76$ and the last $28\times 4-1=111$ bits (for the last step, we do not need to end with a large list, but only with an interesting element, hence we can constrain more). We bound the success probability by the success probability of one complete merging (greater than 1/3) times the success probability of the quantum Fourier Transform (greater than $\pi^2/4$), for a total probability greater than 1/8.

The cost in memory is of 2^{30}, as we store at most 4 lists of size 2^{28}. For the number of quantum queries: there are $4^3 = 64$ initial lists in the tree, each costs 24 queries (to obtain a list of 2^{24} labels by combining). We have to redo this 256 times to obtain all the labels we want, and to repeat this 8 times due to the probability of success. Hence, the query cost is $24 \times 64 \times 256 \times 8 \simeq 2^{22}$. The classical time cost is in $256 \times 8 \times 3 \times 2^{28\times 2} \simeq 2^{69}$. The quantum time cost is in $256 \times 8 \times 3 \times 2^{28} \simeq 2^{41}$.

4 Computing the CSIDH Group Action

The previous section dealt with the cost of quantum abelian hidden shift algorithms, abstracting out the functions f and g that we query in these algorithms. In the context of CSIDH, a "query" means computing the group action, $[x]\cdot E$ with some curve E and an ideal $[x]$, with an arbitrary representation. In this section, we show that this costs only a little more than an evaluation of $[\mathfrak{l}_i]^{x_i}$ with $-m \leq x_i \leq m$.

First of all, we use Shor's algorithm to decompose $[x]$ over the $[\mathfrak{l}_i]$. The decomposition $[x] = \prod_i [\mathfrak{l}_i]^{t_i}$ can contain high exponents t_i. This section deals with reducing these exponents using lattice reduction. This idea was used in [5] to decompose arbitrary elements of $\mathcal{C}\ell(\mathcal{O})$ as products of small-degree isogenies. The main difference with Algorithm 7 in [5] is that the ideal classes upon which we decompose are already given by the scheme, and we want a non-asymptotic bound.

4.1 Finding a Basis of Multi-periods

Given p and the ideal classes $[\mathfrak{l}_1], \ldots, [\mathfrak{l}_u]$, the integer vectors $\bar{e} = (e_1, \ldots e_u)$ such that $[\mathfrak{l}_1]^{e_1} \ldots [\mathfrak{l}_u]^{e_u} = \mathbf{1}$ form an integer lattice in $\mathbb{R}^u$. We note $\mathcal{L}$ this lattice of "multi-periods". First, in a precomputation step, we find a short basis of $\mathcal{L}$.

Computing Relations. Computing a basis of $\mathcal{L}$ means computing the kernel of the map:

$$\begin{aligned} \mathbb{Z}_{\ell_1} \times \ldots \times \mathbb{Z}_{\ell_u} &\to \mathcal{C}\ell(\mathcal{O}) \\ (e_1, \ldots e_u) &\mapsto [\mathfrak{l}_1]^{e_1} \ldots [\mathfrak{l}_u]^{e_u} \end{aligned}$$

So we can just use a quantum Hidden Subgroup algorithm in an abelian group. In other words, we are decomposing the group $\{[\mathfrak{l}_1]^{e_1} \ldots [\mathfrak{l}_u]^{e_u}, e_1, \ldots e_u \in \mathbb{Z}\}$ as a product $\mathbb{Z}_{\ell_1} \times \ldots \times \mathbb{Z}_{\ell_u}$, using the technique of [11]. After these precomputations, we obtain a basis of the lattice $\mathcal{L}$.

4.2 Finding a Short Basis

We compute an approximate short basis using the best known algorithm to date, the Block Korkine Zolotarev algorithm (BKZ) [34]. Its complexity depends on the dimension u and the *block size*, an additional parameter which determines the quality of the basis. Practical complexity analyses can be found in [22] and [10]. The dimension of the lattice $\mathcal{L}$ for CSIDH parameters is between 74 and 209. Even for *a priori* random lattices, this is small, suggesting that even the shortest lattice vector problem could be solved in reasonable time.

Basis Quality. Generally, the quality of the basis is related to the Hermite factor. 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:

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

BKZ-20 gives a heuristic constant c of approximately 1.0128 [22]. Furthermore, in [22, Fig. 12], the authors give a running time for BKZ-20 of the order 1000 CPU seconds for dimension 200.

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 $||b_1||_2 \leq 2c^u m \sqrt{u}$.

4.3 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 $B = b_1, \dots b_u$ of $\mathcal{L}$. Indeed, given a vector $\bar{v} = (v_1 \dots v_u)$ 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 simultaneously for all vectors of $\mathbb{Z}^n$.

We use Babai's nearest-plane algorithm [2] (see *e.g.* [21], chapter 18). Given the target vector $\bar{t}$, a reduced basis B and its Gram-Schmidt orthogonalization

$B^\star$, this algorithm outputs in polynomial time 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 [22] and [10] 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, the L_1 norm counts the number of successive isogenies to be applied. We can count (roughly) the action of $\prod_i [\mathfrak{l}_i]^{t_i - v_i}$ as $||\bar{v} - \bar{t}||_1 / (um)$ equivalent "legitimate" class group actions. The closest we are to the lattice $\mathcal{L}$, the smallest the representation (via $\bar{v} - \bar{t}$) of class group elements becomes. Naturally, if we manage to solve the exact CVP, and obtain always the closest vector to $\bar{t}$, any class group action evaluation will have exponents in $\{-m, \ldots, m\}$ and cost exactly the same as a "legitimate" one. 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 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 $\mathcal{L}$ 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 5 is one of the main components of Algorithm 7 in [5] for faster isogeny evaluations. In a classical context, the authors also use BKZ to reduce the lattice basis 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.

In [6], this algorithm for fast evaluation is further applied to the CSIDH scheme. However, the authors are more focusing on the asymptotic time complexity; and asymptotically, using the ideals $[\mathfrak{l}_i]$ provided by the scheme as a decomposition basis 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.

4.4 Computing the Group Action

We are now ready to compute the action of any $[\mathfrak{g}]$, in three steps:

- Using Shor's algorithm, we decompose $[\mathfrak{g}]$ over the $[\mathfrak{l}_i]$, as $[\mathfrak{g}] = \prod_i [\mathfrak{l}_i]^{t_i}$.
- Using Babai's nearest-plane algorithm with the basis B, we find the approximate closest vector $\bar{v}$: this requires u^2 multiplications of vectors coordinates, which are approximately $\log_2 p$-bit integers;
- We compute the action of $\prod_i [\mathfrak{l}_i]^{t_i - v_i}$.

For each set of parameters, computing the isogenies remains the major cost. 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.

If the Shortest Vector Problem is solvable exactly for lattices from the CSIDH parameters, the cost overhead of the group action with respect to a legitimate key exchange computation becomes $\sqrt{u}$ (with $c = 1$ as approximation factor), hence smaller than 2^4 for all parameters.

Simulations. In practice, we performed simulations by taking a cyclic class group $\mathcal{C}\ell(\mathcal{O})$ of random cardinality $q \simeq \sqrt{p}$. Then we take u elements at random in this group, of the form g^{a_i} for some generator g and compute two-by-two relations between them, as: $(g^{a_i})^{a_{i+1}} \cdot (g^{a_{i+1}})^{-a_i} = \mathbf{1}$. The sparsity of the basis obtained seems to help. The computational system Sage [38] performs BKZ reduction with blocksize 20 in a handful of minutes in dimension 200. We can also directly compute the quantity:

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

which bounds the L_1 norm of the vector in output of Babai's algorithm. We compute its average for some lattices generated as above (the standard deviation from the values found does not exceed 10%). We find an approximation factor in L_1 of less than 2^3 in all three dimensions for Babai's algorithm.

Once (a representation of) the secret isogeny has been recovered, this factor also gives the multiplicative overhead on the size (in L_1) of the representation found against the original one.

Quantum Circuits for the CSIDH Group Action. The number of quantum gates required to evaluate a CSIDH group action, with exponents in the range $-m \dots m$, has been estimated in full detail in [4]. The authors give 765325228976 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 many 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.

This cost is not given in [4] 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 the hidden shift algorithms).

By combining these costs with the approximation factors of 2^3 that we estimated above, we are now able to give an estimation of the number of Clifford + T gates required for an evaluation of $[x] \cdot E$ in superposition over the whole class group $\mathcal{C}\ell(\mathcal{O})$, for a given bit-size of p. This is Table 4.

Table 4: Number of Clifford + T gates required to compute $[x]{\cdot}E$ in superposition over the class group $\mathcal{C}\ell(\mathcal{O})$.

Targeted level in [9]	$\log_2 p$	Number of gates (in $\log_2$)
NIST 1	512	$3 + 48 = 51$
NIST 3	1024	$3 + 53 = 56$
NIST 5	1792	$3 + 56 = 59$

The costs of Table 4 seem high, but they are directly related to the implementation of the scheme. A massive improvement in the quantum implementation of CSIDH, although it seems unlikely, would reduce these costs dramatically, even if the number of queries required by the hidden shift algorithm is unchanged.

5 Estimating the Security of CSIDH Parameters

The parameters in [9] are aimed at three security levels defined by the NIST call [31]: NIST 1 should be as computationally hard as recovering the secret key of AES-128 (with quantum or classical resources), NIST 3 should be as hard as key-recovery of AES-192 and NIST 5 key-recovery of AES-256.

A key-recovery on full 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 requires one or more evaluations of a quantum circuit implementing AES. Such a circuit was designed in [23]. It costs approximately 2^{20} quantum gates of the universal Clifford+T set. Without the MAXDEPTH assumption, the time complexity of running Grover's algorithm for AES-128, 192 and 256 is respectively[5] $2^{85.9}$, $2^{119.1}$ and $2^{151.3}$.

If the first algorithm has a time cost below Grover's algorithm, it uses a large amount of quantum memory. This issue is resolved in the two other approaches, that only need *classical* memory, their only quantum memory requirement being the memory cost of the oracle, plus a polynomial number of qubits.

Table 5: CSIDH attack cost with Algorithm 2 in $\log_2$ scale, compared with the corresponding Grover key-recovery on AES.

Level	Reference AES instance	Grover Cost	Attack quantum time cost	Attack quantum memory cost
NIST 1	AES-128	85.9	83.5	31
NIST 3	AES-192	119.1	100.5	43
NIST 5	AES-256	151.3	114.5	56

Minimal quantum cost. As the quantum queries are very costly in the case of CSIDH, we can first try to use the variant of Section 3.3. The quantum time complexity now falls far below the expected level, but the dominating complexity is the cost of the classical subset-sum instance. We estimate that each instance costs $2^{0.291 \log_2 N}$, dismiss the subset-sum polynomial factor, and compare this cost to the classical query cost of AES exhaustive search, which is a standard approach of security estimates. However, we take into account a factor $8(\log_2 N)$ due to the number of subset-sum instances that have to be solved (one for each label produced before the final QFT, and a success probability of $\frac{1}{8}$ in total). We also count as $8(\log_2 N)^2$ the number of quantum queries to the oracle, where N is the cardinality of the class group (roughly $\sqrt{p}$).

The attack using the subset-sum approach of Section 3.3 already allows to break the parameters proposed for NIST levels 1 and 3, as showed in Table 6. A quantum adversary can find the secret-key using only a million quantum queries to the scheme (respectively 2^{19} and 2^{21}) and less classical computations than what an AES secret-key recovery requires. The quantum memory used is also limited, and depends mostly on the implementation of the CSIDH oracle.

[5] We have reduced the gate counts of [23] by supposing that we use only respectively one, two and two plaintext-ciphertext pairs.

Table 6: CSIDH quantum attack with minimal quantum cost, in $\log_2$ scale.

Targeted level in [9]	Expected classical time	Grover cost	Proposed $\log_2 p$ in [9]	Quantum time of our attack	Classical time/mem. of our attack
NIST 1	128	85.9	512	$19 + 51 = 70$	86
NIST 3	192	119.1	1024	$21 + 56 = 77$	161
NIST 5	256	151.3	1792	$22 + 59 = 81$	274

Classical/Quantum tradeoffs. We can trade between classical and quantum cost with the algorithm of Section 3.4. We summarize in Table 7 the cost with 4-list merging, minimal and equal quantum query and classical memory (excluding polynomial factors). Hence, we considered that we had lists of size $2^{\sqrt{2\log_2(N)/3}}$ everywhere and $\sqrt{\log_2(N)/6}$ steps, and computed the costs accordingly.

Table 7: A possible tradeoff with Kuperberg's algorithm, in $\log_2$ scale.

Level	Expected classical time	Grover time	Attack Quantum time	Attack Class. time	Attack Class. mem
NIST 1	128	85.9	76	51	18
NIST 3	192	119.1	87	69	25
NIST 5	256	151.3	96	87	31

Safe parameters. As we can see, all the proposed instances fall below their targeted security levels. As the algorithms are subexponential and allow for many tradeoffs, the minimal size for an instance to be safe is hard to estimate, and vastly depend on the precise cost metrics. Even if the NIST offer some cost metrics for quantum attacks, there is still some margin of interpretation.

For example, if we neglect the polynomial factors and want an attack to cost at least 2^{128} classical time and 2^{64} quantum time, quantum queries and classical memory, then p should have roughly 6000 bits. A different interpretation may restrict the allowed resources to 2^{64} classical time, quantum time and classical memory and 2^{30} quantum queries (in order to take into account the cost of a single query). In that case, p should have roughly 3000 bits. The optimal algorithm for a given set of resources may involve different techniques, and in order to find the best combination, it may be relevant to use a MILP solver.

6 Conclusion

We performed the first non-asymptotic quantum security assessment of CSIDH, a recent and promising key-exchange primitive based on supersingular elliptic curve isogenies. We presented the main variants of quantum commutative hidden shift algorithms, which are used as a building block in attacking CSIDH. We gave tradeoffs, estimates and experimental simulations of their complexities. Next, we showed that evaluating the oracle in these algorithms adds little overhead to an evaluation of the class group action in CSIDH. Putting together and completing the available literature on quantum hidden shift algorithms, lattice reduction techniques in isogeny-based cryptography, and quantum circuits for evaluating isogenies in a CSIDH context, we were able to propose the first non-asymptotic cost estimates of attacking CSIDH.

These estimates are given as a number of gates in the quantum circuit model and can be compared to the targeted security levels, as defined in the ongoing NIST call. Using different tradeoffs, we showed that the parameters proposed [9] did not meet these levels.

There are many tradeoffs in quantum hidden shift algorithms. This makes the security analysis of CSIDH all the more challenging, and we tried to be as exhaustive as possible regarding the current literature. In particular, the CSIDH-512 instance is roughly 50 000 times easier to break quantumly than AES-128, using a variant *polynomial* in quantum queries and *exponential* in classical computations.

Other Isogeny-based Schemes. The NIST candidate SIKE [15] is not affected, as it uses supersingular elliptic curves on $\mathbb{F}_{p^2}$, for which there is no clear global group action.

The idea of using lattice reduction in order to speed up the computation of the class group action in superposition can be applied to ordinary isogenies, as remarked by multiple authors [16, 9, 5], by setting up an oracle alternative to the one given in [12].

Recently, De Feo, Kieffer and Smith [20] 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. With refinements, the same lattice reduction technique can be adapted to their situation.

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 [5], Luca De Feo, Ben Smith and Steven Galbraith for helpful comments on Kuperberg's algorithm and discussions on the NIST benchmark. Thanks to the anonymous Eurocrypt and Crypto referees for their helpful remarks.

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. Adj, G., Cervantes-Vázquez, D., Chi-Domínguez, J., Menezes, A., Rodríguez-Henríquez, F.: On the cost of computing isogenies between supersingular elliptic curves. In: SAC. Lecture Notes in Computer Science, vol. 11349, pp. 322–343. Springer (2018)
2. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986), https://doi.org/10.1007/BF02579403
3. Becker, A., Coron, J., Joux, A.: Improved generic algorithms for hard knapsacks. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 6632, pp. 364–385. Springer (2011)
4. 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
5. 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)
6. Biasse, J.F., Jacobson, M.J., Iezzi, A.: A note on the security of CSIDH. CoRR (2018), https://arxiv.org/abs/1806.03656
7. Bonnetain, X.: Improved low-qubit hidden shift algorithms. CoRR abs/1901.11428 (2019), http://arxiv.org/abs/1901.11428
8. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 11272, pp. 560–592. Springer (2018)
9. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 11274, pp. 395–427. Springer (2018)
10. 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)
11. 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
12. 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
13. Cohen, H., Lenstra, H.W.: Heuristics on class groups of number fields. In: Number Theory Noordwijkerhout 1983, pp. 33–62. Springer (1984)
14. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
15. De Feo, L., 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
16. De Feo, L., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 11274, pp. 365–394. Springer (2018)

17. Decru, T., Panny, L., Vercauteren, F.: Faster seasign signatures through improved rejection sampling. IACR Cryptology ePrint Archive 2018, 1109 (2018)
18. 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)
19. Feo, L.D., Galbraith, S.D.: Seasign: Compact isogeny signatures from class group actions. IACR Cryptology ePrint Archive 2018, 824 (2018)
20. Feo, L.D., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 11274, pp. 365–394. Springer (2018)
21. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press (2012), https://www.math.auckland.ac.nz/ sgal018/crypto-book/crypto-book.html
22. 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)
23. 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)
24. Jalali, A., Azarderakhsh, R., Kermani, M.M., Jao, D.: Towards optimized and constant-time CSIDH on embedded devices. In: COSADE. Lecture Notes in Computer Science, vol. 11421, pp. 215–231. Springer (2019)
25. Jaques, S., Schanck, J.M.: Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. IACR Cryptology ePrint Archive 2019, 103 (2019)
26. 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
27. 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
28. Meyer, M., Campos, F., Reith, S.: On lions and elligators: An efficient constant-time implementation of csidh. Cryptology ePrint Archive, Report 2018/1198 (2018), https://eprint.iacr.org/2018/1198
29. Meyer, M., Reith, S.: A faster way to the CSIDH. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 11356, pp. 137–152. Springer (2018)
30. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information. AAPT (2002)
31. 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
32. Peikert, C.: He gives c-sieves on the csidh. Cryptology ePrint Archive, Report 2019/725 (2019), https://eprint.iacr.org/2019/725

33. Regev, O.: A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space. CoRR (2004), http://arxiv.org/abs/quant-ph/0406151
34. 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
35. Schroeppel, R., Shamir, A.: A t=o($2^{n/2}$), s=o($2^{n/4}$) algorithm for certain np-complete problems. SIAM J. Comput. 10(3), 456–464 (1981), https://doi.org/10.1137/0210033
36. 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
37. Stolbunov, A.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. Adv. in Math. of Comm. 4(2), 215–235 (2010), https://doi.org/10.3934/amc.2010.4.215
38. The Sage Developers: SageMath, the Sage Mathematics Software System, `http://www.sagemath.org`