J. Math. Cryptol. **8** (2014), 1–29
DOI 10.1515/jmc-2012-0016 © de Gruyter 2014

Constructing elliptic curve isogenies in quantum subexponential time

Andrew Childs, David Jao and Vladimir Soukharev

Communicated by María González Vasco

Abstract. Given two ordinary elliptic curves over a finite field having the same cardinality and endomorphism ring, it is known that the curves admit a nonzero isogeny between them, but finding such an isogeny is believed to be computationally difficult. The fastest known classical algorithm takes exponential time, and prior to our work no faster quantum algorithm was known. Recently, public-key cryptosystems based on the presumed hardness of this problem have been proposed as candidates for post-quantum cryptography. In this paper, we give a new subexponential-time quantum algorithm for constructing nonzero isogenies between two such elliptic curves, assuming the Generalized Riemann Hypothesis (but with no other assumptions). Our algorithm is based on a reduction to a hidden shift problem, and represents the first nontrivial application of Kuperberg's quantum algorithm for finding hidden shifts. This result suggests that isogeny-based cryptosystems may be uncompetitive with more mainstream quantum-resistant cryptosystems such as lattice-based cryptosystems. As part of this work, we also present the first classical algorithm for evaluating isogenies having provably subexponential running time in the cardinality of the base field under GRH.

Keywords. Elliptic curves, isogenies, hidden shift problem, quantum algorithms.

2010 Mathematics Subject Classification. 81P94, 68Q12, 11Y40, 14H52.

1 Introduction

We consider the problem of constructing a nonzero isogeny between two given isogenous ordinary elliptic curves defined over a finite field $\mathbb{F}_q$ and having the same endomorphism ring. This problem has led to several applications in elliptic curve cryptography, both constructive and destructive. The fastest known probabilistic algorithm for solving this problem is that of Galbraith and Stolbunov [14], based on the work of Galbraith, Hess, and Smart [13]. Their algorithm is exponential, with a worst-case (and average-case) running time roughly proportional to $\sqrt[4]{q}$.

This work was supported in part by MITACS, NSERC, the Ontario Ministry of Research and Innovation, QuantumWorks, and the US ARO/DTO.

Although quantum attacks are known against several cryptographic protocols of an algebraic nature [11, 15, 28], until now there has been no nontrivial quantum algorithm for constructing isogenies. The difficulty of this problem has led to various constructions of public-key cryptosystems based on finding isogenies. The first such proposal appears in a preprint of Couveignes [9], although it makes no mention of quantum computation. More recently, Rostovtsev and Stolbunov [25] and Stolbunov [30] proposed refined versions of these cryptosystems with the specific aim of obtaining cryptographic protocols that resist attacks by quantum computers.

In this work, we give a subexponential-time quantum algorithm for constructing a nonzero isogeny between two given elliptic curves (of the type arising in the aforementioned cryptosystems). We show that the running time of our algorithm is bounded above by $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$ under (only) the Generalized Riemann Hypothesis (GRH), where

$$L_N(\alpha, c) := \exp\big[(c + o(1))(\ln N)^{\alpha}(\ln \ln N)^{1-\alpha}\big].$$

This result raises serious questions about the viability of isogeny-based cryptosystems in the context of quantum computers. At present, isogeny-based cryptosystems are not especially attractive since their performance is poor compared to other quantum-resistant cryptosystems, such as lattice-based cryptography [16]. Nevertheless, they represent a distinct family of cryptosystems worthy of analysis (for reasons of diversity if nothing else, given the small number of quantum-resistant public-key cryptosystem families available [23]). Since isogeny-based cryptosystems already perform poorly at moderate security levels [30, Table 1], any improved attacks such as ours would seem to disqualify such systems from consideration when the possibility of efficient quantum computation is taken into account.

1.1 Contributions

Our first main contribution, described in Section 4, is a reduction from the problem of isogeny construction to the abelian hidden shift problem. While a connection between isogenies and hidden shifts was noted previously by Stolbunov [30], we make the simple observation that the reduction gives an *injective* hidden shift problem. This lets us apply an algorithm of Kuperberg [21] to solve the hidden shift problem using a subexponential number of queries to certain functions. Though straightforward, this reduction constitutes the first nontrivial application of Kuperberg's algorithm outside of the black-box setting.

The reduction to the hidden shift problem implies that a subexponential-time algorithm for computing the hiding functions yields a subexponential-time algorithm for computing isogenies. Although such subexponential-time algorithms

were previously known [19], their running time analysis depends on nonstandard heuristic assumptions. Our second main contribution, described in Section 3, is a subexponential-time (classical) algorithm to compute the hiding functions whose running time analysis depends only on GRH. We achieve this improvement using expansion properties of a certain Cayley graph [18].

Kuperberg's algorithm for the abelian hidden shift problem uses superpolynomial space (i.e., a quantum computer with superpolynomially many qubits), so the same is true of the most straightforward version of our algorithm. Since it is difficult to build quantum computers with many qubits, this feature could limit the applicability of our result. However, we also obtain an algorithm using polynomial space by taking advantage of Regev's alternative approach to the abelian hidden shift problem [24]. Regev only explicitly considered the case of the hidden shift problem in a cyclic group whose order is a power of 2, and even in that case did not compute the constant in the exponent of the running time. We fill both of these gaps in our work, showing that the hidden shift problem in any finite abelian group A can be solved in time $L_{|A|}(\frac{1}{2}, \sqrt{2})$ by a quantum computer using only polynomial space. Consequently, we give a polynomial-space quantum algorithm for isogeny construction using time $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2} + \sqrt{2})$. The group relevant to isogeny construction is not always cyclic, so the extension to general abelian groups is necessary for our application.

1.2 Related work

Our algorithm for evaluating the hiding functions is based on reducing an ideal modulo principal ideals to obtain a smooth ideal. This idea is originally due to Galbraith, Hess, and Smart [13]. Bröker, Charles, and Lauter [6] and Jao and Soukharev [19] also use this idea to give algorithms for evaluating isogenies. Bisson and Sutherland [5] use a similar smoothing technique to compute endomorphism rings in subexponential time. We stress that, with the exception of [6], which is restricted in scope to small discriminants, all the results mentioned above make heuristic assumptions of varying severity [5, §4], [13, p. 37], [19, p. 224] in addition to the Generalized Riemann Hypothesis in the course of proving their runtime claims. Our work is the first to achieve provably subexponential running time with no heuristic assumptions other than GRH. In practice, the heuristic algorithms in [5] and [19] run slightly faster than our algorithms in Section 3 – though their asymptotic running times are identical – because they make use of an optimized exponent distribution (originating from [5]) that minimizes the number of large-degree isogenies appearing in the smooth factorization. We do not use this optimization, because doing so would reintroduce the need for additional heuristic assumptions.

Following the appearance of our work, Bisson [4, Theorem 6.1] gave a subexponential algorithm for computing endomorphism rings of ordinary elliptic curves under only GRH, citing Theorem 2.1 of our work in the proof. Bisson also presents a faster algorithm [4, Proposition 4.4] for directly determining the isogenous curve corresponding to a prime ideal, which can be used to improve our algorithm (see Remark 3.4).

An alternative approach to computing isogenies, considered by Couveignes [9, p. 11] and Stolbunov [30, p. 227], is to treat the class group as a $\mathbb{Z}$-module and use lattice basis reduction to compute the action of the class group on elliptic curves. In practice, the lattice-based approach works well for moderate parameter sizes. However, since it amounts to solving the closest vector problem, the method asymptotically requires exponential time (even with known quantum algorithms), and thus is slower than our approach.

2 Isogenies

For general background on elliptic curves, we refer the reader to Silverman [29].

Let E and E' be elliptic curves defined over a field F. An *isogeny* $\phi\colon E \to E'$ is an algebraic morphism mapping the identity element of E to the identity element of E'. For consistency with the definition of Silverman [29, Section III.4], which permits zero isogenies, we must include the zero isogeny in our definition. However, throughout the paper, when we refer to computing or constructing isogenies, we always mean the computation or construction of a nonzero isogeny. The *degree* of an isogeny is its degree as an algebraic morphism. The *endomorphism ring* $\mathrm{End}(E)$ is the set of isogenies from E to itself over $\bar{F}$. This set forms a ring under pointwise addition and composition.

When F is a finite field, the rank of $\mathrm{End}(E)$ as a $\mathbb{Z}$-module is either 2 or 4. We say E is *supersingular* if the rank is 4, and *ordinary* otherwise. A supersingular curve cannot be isogenous to an ordinary curve. In this paper, as in [30], we restrict our attention to ordinary elliptic curves. Our results have motivated a separate study of cryptographic protocols based on isogenies between supersingular curves [17], and it remains an interesting open problem to better understand the computational difficulty of constructing such isogenies in the supersingular case.

Over a finite field $\mathbb{F}_q$, two elliptic curves E and E' are isogenous if and only if $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$; see [31]. The endomorphism ring of an ordinary elliptic curve over a finite field is an imaginary quadratic order $\mathcal{O}_\Delta = \mathbb{Z}[\frac{\Delta+\sqrt{\Delta}}{2}]$ of discriminant $\Delta < 0$. In general, all curves over $\mathbb{F}_q$ with the same endomorphism ring are isogenous up to a twist. The set of all isomorphism classes (over $\bar{\mathbb{F}}_q$) of isogenous curves with endomorphism ring $\mathcal{O}_\Delta$ is denoted $\mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$, where n is

the cardinality of any such curve. We represent elements of $\mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ by taking the j-invariant of any representative curve in the isomorphism class.

Any separable isogeny $\phi\colon E \to E'$ between curves in $\mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ can be specified, up to isomorphism, by giving E and $\ker\phi$; see [29, Proposition III.4.12]. The kernel of an isogeny, in turn, can be represented as an ideal in $\mathcal{O}_\Delta$; see [32, Theorem 4.5]. Denote by $\phi_{\mathfrak{b}}\colon E \to E_{\mathfrak{b}}$ the isogeny corresponding to an ideal $\mathfrak{b}$ (keeping in mind that $\phi_{\mathfrak{b}}$ is only defined up to isomorphism of $E_{\mathfrak{b}}$). Principal ideals correspond to endomorphisms, so any other ideal equivalent to $\mathfrak{b}$ in the ideal class group $\mathrm{Cl}(\mathcal{O}_\Delta)$ of $\mathcal{O}_\Delta$ yields the same codomain curve $E_{\mathfrak{b}}$, up to isomorphism [32, Theorem 3.11]. Hence one obtains a well-defined group action $*\colon \mathrm{Cl}(\mathcal{O}_\Delta) \times \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta) \to \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ taking $[\mathfrak{b}] * j(E)$ to $j(E_{\mathfrak{b}})$, where $[\mathfrak{b}]$ denotes the ideal class of $\mathfrak{b}$. This group action, which we call the *complex multiplication action*, is free and transitive [32, Theorem 4.5], and thus $\mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ forms a principal homogeneous space over $\mathrm{Cl}(\mathcal{O}_\Delta)$.

2.1 Isogeny graphs under GRH

Our runtime analysis in Section 3 relies on the following result of [18] stating, roughly, that random short products of small primes in $\mathrm{Cl}(\mathcal{O}_\Delta)$ yield nearly uniformly random elements of $\mathrm{Cl}(\mathcal{O}_\Delta)$, under GRH.

Theorem 2.1. *Let $\mathcal{O}_\Delta$ be an imaginary quadratic order of discriminant $\Delta < 0$ and conductor c. Set $G = \mathrm{Cl}(\mathcal{O}_\Delta)$. Let B and x be real numbers satisfying $B > 2$ and $x \geq (\ln|\Delta|)^B$. Let S_x be the multiset $A \cup A^{-1}$, where*

$$A = \left\{[\mathfrak{p}] \in G : \gcd(c, \mathfrak{p}) = 1 \text{ and } N(\mathfrak{p}) \leq x \text{ is prime}\right\}$$

with $N(\mathfrak{p})$ denoting the norm of $\mathfrak{p}$. Then, assuming GRH, there exists a positive absolute constant $C > 1$, depending only on B, such that for all Δ, a random walk of length

$$t \geq C \frac{\ln|G|}{\ln\ln|\Delta|}$$

in the Cayley graph $\mathrm{Cay}(G, S_x)$ from any starting vertex lands in any fixed subset $S \subset G$ with probability at least $\frac{1}{2}\frac{|S|}{|G|}$.

Proof. Apply [18, Corollary 1.3] with the parameters

- $K =$ the field of fractions of $\mathcal{O}_\Delta$,
- $G = \mathrm{Cl}(\mathcal{O}_\Delta)$,
- $q = |\Delta|$.

Following [10], we refer to $G = \mathrm{Cl}(\mathcal{O}_\Delta)$ as the *ring class group* of Δ. Observe that by [18, Remark 1.2 (a)], Corollary 1.3 of [18] applies to the ring class group G, since ring class groups are quotients of narrow ray class groups [10, p. 160]. By [18, Corollary 1.3], Theorem 2.1 holds for all sufficiently large values of $|\Delta|$, i.e., for all but finitely many $|\Delta|$. To prove the theorem for all $|\Delta|$, simply take a larger (but still finite) value of C. □

Corollary 2.2. *Theorem* 2.1 *still holds with the set A redefined as*

$$A = \big\{[\mathfrak{p}] \in G : \gcd(m\Delta, \mathfrak{p}) = 1 \text{ and } N(\mathfrak{p}) \leq x \text{ is prime}\big\},$$

where m is any integer having at most $O(x^{1/2-\varepsilon} \log|\Delta|)$ *prime divisors.*

Proof. The alternative definition of the set A differs from the original definition by no more than $O(x^{1/2-\varepsilon} \log|\Delta|)$ primes. As stated in [18, p. 1497], the contribution of these primes can be absorbed into the error term $O(x^{1/2} \log(x) \log(xq))$, and hence does not affect the conclusion of Theorem 2.1. □

2.2 The group action inverse problem

For a fixed discriminant Δ, the *vectorization* [9, §2] or *group action inverse* [30, §2.4] problem is the task of finding an ideal class $[\mathfrak{b}] \in \mathrm{Cl}(\mathcal{O}_\Delta)$ such that $[\mathfrak{b}] * j(E) = j(E')$, given $j(E)$ and $j(E')$. We refer to $[\mathfrak{b}]$ as the *quotient* of $j(E)$ and $j(E')$. The computational infeasibility of finding quotients in $\mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ is a necessary condition for the security of isogeny-based cryptosystems [9, §3], [30, §7]. In the remainder of this paper, we present our subexponential algorithm for evaluating quotients in $\mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ on a quantum computer.

A notable property of isogeny-based cryptosystems is that they do not require the ability to evaluate the complex multiplication action efficiently on arbitrary inputs. It is enough to sample from random smooth ideals (for which $*$ can be evaluated efficiently) when performing operations such as key generation [9, §5.4], [30, §6.2]. However, to attack these cryptosystems using our approach, we *do* require the ability to evaluate the complex multiplication action on arbitrary inputs. We turn to this problem in the next section.

3 Computing the complex multiplication action

In this section, we describe a new classical (i.e., non-quantum) algorithm to evaluate the complex multiplication action and show that, under GRH, our algorithm has a running time of $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$, which is subexponential in the input size. All notation is as in Section 2. Given an ideal class $[\mathfrak{b}]$ in $\mathrm{Cl}(\mathcal{O}_\Delta)$, and a j-invariant

$j(E)$ of an ordinary elliptic curve E of endomorphism ring $\mathcal{O}_\Delta$ over $\mathbb{F}_q$, we wish to evaluate $[\mathfrak{b}] * j(E)$. Recall that

$$L_N(\tfrac{1}{2}, c) := \exp\big[(c + o(1))\sqrt{\ln N \ln\ln N}\big].$$

For convenience, we denote $L_{\max\{|\Delta|,q\}}(\frac{1}{2}, c)$ by $L(c)$.

Our algorithms are modified versions of prior algorithms that also achieved asymptotically identical subexponential running time, but under additional heuristics. Algorithm 1 is based on [19, Algorithm 3], which is in turn based on Seysen's algorithm [27]; Algorithm 2 is based on [6, Algorithm 4.1]. Our bounds on t in Algorithm 1 are new, and allow us to prove the crucial runtime bound (Proposition 3.1).

Computing a relation. Given an ideal class $[\mathfrak{b}] \in \mathrm{Cl}(\mathcal{O}_\Delta)$, Algorithm 1 produces a relation vector $\mathbf{z} = (z_1, \ldots, z_f) \in \mathbb{Z}^f$ for $[\mathfrak{b}]$, with respect to a factor base $\mathcal{F} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_f\}$, satisfying $[\mathfrak{b}] = \mathcal{F}^{\mathbf{z}} := \mathfrak{p}_1^{z_1} \cdots \mathfrak{p}_f^{z_f}$, with the additional property (cf. Proposition 3.2) that the L^1-norm $|\mathbf{z}|_1$ of $\mathbf{z}$ is less than $O(\ln|\Delta|)$ for some absolute implied constant (here the L^1 norm of a vector is the sum of the absolute values of its coordinates). Algorithm 1 is similar to [7, Algorithm 11.2], except that we impose a constraint on $|\mathbf{v}|_1$ in Step 5 in order to keep $|\mathbf{z}|_1$ small, and (for performance reasons) we use Bernstein's algorithm instead of trial division to find smooth elements. Alternatively, one can use Lenstra's elliptic curve method, which reduces the space requirement from superpolynomial to polynomial. However, the running time analysis of that method requires additional heuristics, which we are trying to avoid. (On a quantum computer, there are no such difficulties: one can simply factor integers directly in polynomial time [28].)

Corollary 9.3.12 of [7] together with the restriction $C > 1$ in Theorem 2.1 implies that there exists a value of t satisfying the inequality in Algorithm 1.

Computing $j(E')$. Algorithm 2 is the main algorithm for evaluating the complex multiplication action. It takes as input a discriminant $\Delta < 0$, an ideal class $[\mathfrak{b}] \in \mathrm{Cl}(\mathcal{O}_\Delta)$, and a j-invariant $j(E) \in \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$, and produces as output the element $j(E') \in \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ such that $[\mathfrak{b}] * j(E) = j(E')$. Primes dividing $q \cdot n \cdot \Delta$ must be eliminated in order to compute the isogenies in the final step of the algorithm (cf. [6, Algorithm 4.1]). In Step 2 of the algorithm, we adopt the same convention used in [6, p. 102], where the notation $E[\mathfrak{L}]$ for an ideal $\mathfrak{L} \subset \mathrm{End}(E)$ denotes

$$E[\mathfrak{L}] := \big\{P \in E : \alpha(P) = 0 \text{ for all } \alpha \in \mathfrak{L}\big\}.$$

Algorithm 2 is correct since the ideals $\mathfrak{b}$ and $\mathcal{F}^{\mathbf{z}}$ belong to the same ideal class, and thus act identically on $\mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$.

Algorithm 1 Computing a relation

Input: Δ, q, n, z, $[\mathfrak{b}]$, and an integer t satisfying $C\frac{\ln|\mathrm{Cl}(\mathcal{O}_\Delta)|}{\ln\ln|\Delta|} \leq t \leq C\ln|\Delta|$, where C is the constant of Theorem 2.1/Corollary 2.2

Output: A relation vector $\mathbf{z} \in \mathbb{Z}^f$ such that $[\mathfrak{b}] = [\mathcal{F}^{\mathbf{z}}]$, or `nil`

1: Compute a factor base $\mathcal{F} = \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_f\}$ consisting of split primes in $\mathcal{O}_\Delta$ of norm less than $\exp(z\sqrt{\ln N \ln\ln N})$, discarding any primes dividing $q\cdot n\cdot\Delta$
2: Set $\mathcal{S} \leftarrow \emptyset$, $\mathcal{P} \leftarrow \{N(\mathfrak{p}) : \mathfrak{p} \in \mathcal{F}\}$
3: Set $\ell \leftarrow \exp(\frac{1}{4z}\sqrt{\ln N \ln\ln N})$
4: **for** $i = 0$ to ℓ **do**
5: Select $\mathbf{v} \in \{0, \ldots, |\Delta| - 1\}^f$ uniformly at random subject to the condition $|\mathbf{v}|_1 = t$
6: Calculate the reduced ideal $\mathfrak{a}_{\mathbf{v}}$ in the ideal class $[\mathfrak{b}]\cdot[\mathcal{F}^{\mathbf{v}}]$
7: Set $\mathcal{S} \leftarrow \mathcal{S} \cup N(\mathfrak{a}_{\mathbf{v}})$
8: **end for**
9: Using Bernstein's algorithm [3], find a $\mathcal{P}$-smooth element $N(\mathfrak{a}_{\mathbf{v}}) \in \mathcal{S}$ (if one exists), or else return `nil`
10: Find the prime factorization of the integer $N(\mathfrak{a}_{\mathbf{v}})$
11: Using Theorem 3.1 of Seysen [27] on the prime factorization of $N(\mathfrak{a}_{\mathbf{v}})$, factor the ideal $\mathfrak{a}_{\mathbf{v}}$ over $\mathcal{F}$ to obtain $\mathfrak{a}_{\mathbf{v}} = \mathcal{F}^{\mathbf{a}}$ for some $\mathbf{a} \in \mathbb{Z}^f$
12: Return $\mathbf{z} = \mathbf{a} - \mathbf{v}$

3.1 Runtime analysis

Here we determine the asymptotic running time of Algorithm 2, as well as the optimal value of the parameter z in Algorithm 1. As is typical for subexponential-time factorization algorithms involving a factor base, these two quantities depend on each other, so both are calculated simultaneously.

Proposition 3.1. *Under GRH, the probability that a single iteration of the* ***for*** *loop of Algorithm* 1 *produces an $\mathcal{F}$-smooth ideal $\mathfrak{a}_{\mathbf{v}}$ is at least $L(-\frac{1}{4z})$.*

Proof. We adopt the notation used in Theorem 2.1 and Corollary 2.2. Apply Corollary 2.2 with the values $m = qn$, $B = 3$, and $x = f = L(z) \gg (\ln|\Delta|)^B$. Observe that m has at most $O(\log q)$ prime divisors, and

$$O(\log q) \ll L_q(\tfrac{1}{2}, z(\tfrac{1}{2} - \varepsilon)) \leq L(z(\tfrac{1}{2} - \varepsilon)) = x^{1/2-\varepsilon}.$$

Therefore Corollary 2.2 applies. The ideal class $[\mathfrak{b}]\cdot[\mathcal{F}^{\mathbf{v}}]$ is equal to the ideal class obtained by taking the walk of length t in the Cayley graph $\mathrm{Cay}(G, S_x)$, having initial vertex $[\mathfrak{b}]$, and whose edges correspond to the nonzero coordinates of the

Algorithm 2 Computing $j(E')$

Input: Δ, q, $[\mathfrak{b}]$, and a j-invariant $j(E) \in \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$
Output: The element $j(E') \in \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ such that $[\mathfrak{b}] * j(E) = j(E')$

1: Using Algorithm 1 with any valid choice of t, compute a relation $\mathbf{z} \in \mathbb{Z}^f$ such that $[\mathfrak{b}] = [\mathcal{F}^{\mathbf{z}}] = [\mathfrak{p}_1^{z_1}\mathfrak{p}_2^{z_2}\cdots\mathfrak{p}_f^{z_f}]$
2: Compute a sequence of isogenies $(\phi_1, \ldots, \phi_s)$ such that the composition $\phi_c\colon E \to E_c$ of the sequence has kernel $E[\mathfrak{p}_1^{z_1}\mathfrak{p}_2^{z_2}\cdots\mathfrak{p}_f^{z_f}]$, using the method of [6, §3]
3: Return $j(E_c)$

vector $\mathbf{v}$. Hence a random choice of vector $\mathbf{v}$ under the constraints of Algorithm 1 yields the same probability distribution as a random walk in $\mathrm{Cay}(G, S_x)$ starting from $[\mathfrak{b}]$.

Let S be the set of reduced ideals in G with $L(z)$-smooth norm. By [7, Lemma 11.4.4],

$$|S| \geq \sqrt{|\Delta|}L_{|\Delta|}(\tfrac{1}{2}, -\tfrac{1}{4z}) \geq \sqrt{|\Delta|}L(-\tfrac{1}{4z}).$$

Hence, by Corollary 2.2, the probability that $\alpha_{\mathbf{v}}$ lies in S is at least

$$\frac{1}{2}\frac{|S|}{|G|} \geq \frac{1}{2} \cdot \frac{\sqrt{|\Delta|}}{|G|} \cdot L(-\tfrac{1}{4z}).$$

Finally, [7, Theorem 9.3.11] states that $\frac{\sqrt{|\Delta|}}{|G|} \geq \frac{1}{\ln|\Delta|}$. Hence the probability that $\alpha_{\mathbf{v}}$ is $\mathcal{F}$-smooth is at least

$$\frac{1}{2} \cdot \frac{1}{\ln|\Delta|} \cdot L(-\tfrac{1}{4z}) = L(-\tfrac{1}{4z}),$$

as desired. □

The following proposition shows that the relation vector $\mathbf{z}$ produced by Algorithm 1 is guaranteed to have small coefficients.

Proposition 3.2. *Any vector* $\mathbf{z}$ *output by Algorithm* 1 *satisfies* $|\mathbf{z}|_1 < (C+1)\ln|\Delta|$.

Proof. Since $\mathbf{z} = \mathbf{a} - \mathbf{v}$, we have $|\mathbf{z}|_1 \leq |\mathbf{a}|_1 + |\mathbf{v}|_1$. But $|\mathbf{v}|_1 \leq C\ln|\Delta|$ by construction, and the norm of $\alpha_{\mathbf{v}}$ is less than $\sqrt{|\Delta|/3}$; see [7, Proposition 9.1.7]. So $|\mathbf{a}|_1 < \log_2\sqrt{|\Delta|/3} < \log_2\sqrt{|\Delta|} < \ln|\Delta|$. □

Finally, we analyze the running time of Algorithm 2.

Theorem 3.3. *Under GRH, Algorithm* 2 *has a worst-case running time of at most* $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$ *and succeeds with probability at least* $1 - \frac{1}{e}$.

Proof. Algorithm 1 has a running time of $L(z) + L(\frac{1}{4z})$ (dominated by the $b(\log_2 b)^{2+\varepsilon}$ cost of Bernstein's algorithm, where $b = L(z) + L(\frac{1}{4z})$ is the combined size of $\mathcal{S}$ and $\mathcal{P}$) and success probability at least $1-\frac{1}{e}$ (since it loops through ℓ vectors $\mathbf{v}$, each with an independent $1/\ell$ chance of producing a smooth ideal $\mathfrak{a}_{\mathbf{v}}$ by Proposition 3.1). Assuming that it succeeds, the analysis of [19, §4.4] applied to Algorithm 2, together with Propositions 3.1 and 3.2, shows that the running time of Step 2 is at most $L(\frac{1}{4z} + 3z)$. Using the inequality $|\Delta| \leq 4q$, the optimal choice of $z = \frac{1}{2\sqrt{3}}$ yields the running time bound of $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$. □

Remark 3.4. Using our Theorem 2.1 (cited as [4, Theorem 6.1]), Bisson has recently developed a subexponential-time algorithm for determining endomorphism rings of elliptic curves, assuming only GRH. As part of that work, Bisson presents a faster algorithm [4, Proposition 4.4] for determining the curves appearing in the sequence of isogenies in Step 2 of Algorithm 2, with running time quadratic in the isogeny degrees, improving upon the cubic time required in prior algorithms. Using this algorithm, the running time of Algorithm 2 improves to $L_q(\frac{1}{2}, \frac{1}{\sqrt{2}})$.

4 A quantum algorithm for constructing isogenies

Our quantum algorithm for constructing isogenies uses a simple reduction to the abelian hidden shift problem. This problem is defined as follows. Let A be a known finite abelian group (with the group operation written multiplicatively) and let $f_0, f_1 \colon A \to S$ be black-box functions, where S is a known finite set. We say that f_0, f_1 *hide* a shift $s \in A$ if f_0 is injective and $f_1(x) = f_0(xs)$ (i.e., f_1 is a shifted version of f_0). The goal of the hidden shift problem is to determine s using queries to such black-box functions. Note that this problem is equivalent to the hidden subgroup problem in the A-dihedral group, the nonabelian group $A \rtimes \mathbb{Z}_2$, where $\mathbb{Z}_2$ acts on A by inversion.

Isogeny construction is easily reduced to the hidden shift problem using the group action defined in Section 2. Given two isogenous curves E_0, E_1 with endomorphism ring $\mathcal{O}_\Delta$, we define functions $f_0, f_1 \colon \mathrm{Cl}(\mathcal{O}_\Delta) \to \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ that hide $[\mathfrak{s}] \in \mathrm{Cl}(\mathcal{O}_\Delta)$, where $[\mathfrak{s}]$ is the ideal class such that $[\mathfrak{s}] * j(E_0) = j(E_1)$. Specifically, let $f_i([\mathfrak{b}]) = [\mathfrak{b}] * j(E_i)$. Then f_0 and f_1 hide $[\mathfrak{s}]$:

Lemma 4.1. *The function* f_0 *is injective and* $f_1([\mathfrak{b}]) = f_0([\mathfrak{b}][\mathfrak{s}])$.

Proof. Since $*$ is a group action,

$$f_1([\mathfrak{b}]) = [\mathfrak{b}] * j(E_1) = [\mathfrak{b}] * ([\mathfrak{s}] * j(E_0)) = ([\mathfrak{b}][\mathfrak{s}]) * j(E_0) = f_0([\mathfrak{b}][\mathfrak{s}]).$$

If there are distinct ideal classes $[\mathfrak{b}], [\mathfrak{b}']$ such that $f_0([\mathfrak{b}]) = f_0([\mathfrak{b}'])$, then $[\mathfrak{b}] * j(E_0) = [\mathfrak{b}'] * j(E_0)$, which contradicts the fact that the action is free and transitive [32, Theorem 4.5]. Thus f_0 is injective. □

Note that the connection between isogenies and hidden shift problems was described in [30, Section 7.2]. However, that paper did not exploit the connection, and in particular, did not mention the injectivity of the hiding functions in the context of the reduction. Without the assumption that f_0 is injective, the hidden shift problem can be as hard as the search problem, requiring exponentially many queries [2]. With injective hiding functions, the problem has polynomial quantum query complexity [12], allowing for the possibility of faster quantum algorithms.

This reduction allows us to apply quantum algorithms for the hidden shift problem to construct isogenies. The (injective) hidden shift problem can be solved in quantum subexponential time assuming we can evaluate the group action in subexponential time. The latter is possible due to Algorithm 2.

We consider two different approaches to solving the hidden shift problem in subexponential time on a quantum computer. The first, due to Kuperberg [21], has a faster running time but requires superpolynomial space. The second approach generalizes an algorithm of Regev [24]. It uses only polynomial space, but is slower than Kuperberg's original algorithm.

Method 1: Kuperberg's algorithm. Kuperberg's approach to the abelian hidden shift problem is based on the idea of performing a Clebsch–Gordan sieve on coset states.

Theorem 4.2 ([21]). *The abelian hidden shift problem has a quantum algorithm with time and query complexity $2^{O(\sqrt{n})}$, where n is the length of the output, uniformly for all finitely generated abelian groups.*

In our context, $2^{O(\sqrt{n})} = 2^{O(\sqrt{\ln|\Delta|})}$ since $|\mathrm{Cl}(\mathcal{O}_\Delta)| = O(\sqrt{\Delta}\ln\Delta)$; see [7, Theorem 9.3.11]. Furthermore, $2^{O(\sqrt{\ln|\Delta|})} = L(o(1)) = L(0)$ regardless of the value of the implied constant in the exponent, since the exponent on the left has no $\sqrt{\ln\ln|\Delta|}$ term, whereas $L(0)$ does. As mentioned above, Kuperberg's algorithm also requires superpolynomial space (specifically, it uses $2^{O(\sqrt{n})}$ qubits).

Method 2: Regev's algorithm. Regev [24] showed that a variant of Kuperberg's sieve leads to a slightly slower algorithm using only polynomial space. In partic-

ular, he proved Theorem 4.3 below in the case where A is a cyclic group whose order is a power of 2 (without giving an explicit value for the constant in the exponent). Theorem 4.3 generalizes Regev's algorithm to arbitrary finite abelian groups. A detailed proof of the following appears in Section 5 (see Theorem 5.8).

Theorem 4.3. *Let A be a finite abelian group and let functions f_0, f_1 hide some unknown $s \in A$. Then there is a quantum algorithm that finds s with time and query complexity $L_{|A|}(\frac{1}{2}, \sqrt{2})$ using space* $\mathrm{poly}(\log|A|)$.

We now return to the original problem of constructing isogenies. Note that to use the hidden shift approach, the group structure of $\mathrm{Cl}(\mathcal{O}_\Delta)$ must be known. Given Δ, it is straightforward to compute $\mathrm{Cl}(\mathcal{O}_\Delta)$ using existing quantum algorithms (see the proof of Theorem 4.5). Thus, we assume for simplicity that the discriminant Δ is given as part of the input. This requirement poses no difficulty, since all existing proposals for isogeny-based public-key cryptosystems [9, 25, 30] stipulate that $\mathcal{O}_\Delta$ is a maximal order, in which case its discriminant can be computed easily: simply calculate the trace $t(E)$ of the curve using Schoof's algorithm [26], and factor $t(E)^2 - 4q$ to obtain the fundamental discriminant Δ (note of course that factoring is easy on a quantum computer [28]).

Remark 4.4. One can conceivably imagine a situation where one is asked to construct an isogeny between two given isogenous curves of unknown but identical endomorphism ring. Although we are not aware of any cryptographic applications of this scenario, it presents no essential difficulty. Bisson has shown using Theorem 2.1 (see [4, Theorem 6.1]) that the discriminant Δ of any ordinary elliptic curve can be computed in $L_q(\frac{1}{2}, \frac{1}{\sqrt{2}})$ time under only GRH (assuming that factoring is easy, which is the case for quantum computers [28]).

Assuming Δ is known, we decompose $\mathrm{Cl}(\mathcal{O}_\Delta)$ as a direct sum of cyclic groups, with a known generator for each, and then solve the hidden shift problem. The overall procedure is described in Algorithm 3.

Theorem 4.5. *Assuming GRH, Algorithm* 3 *runs in time $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$ (respectively, $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2} + \sqrt{2})$) using Theorem* 4.2 *(respectively, Theorem* 4.3*) to solve the hidden shift problem.*

Proof. We perform Step 1 using [8, Algorithm 10], which determines the structure of an abelian group given a generating set and a unique representation for the group elements. We represent the elements uniquely using reduced quadratic forms, and we use the fact that, under ERH (and hence GRH), the set of ideal classes of norm at most $12\ln^2|\Delta|$ forms a generating set [1, Theorem 4]. Note that the result in

Algorithm 3 Isogeny construction

Input: A finite field $\mathbb{F}_q$, a discriminant $\Delta < 0$, and Weierstrass equations of isogenous elliptic curves E_0, E_1 with endomorphism ring $\mathcal{O}_\Delta$

Output: $[\mathfrak{s}] \in \mathrm{Cl}(\mathcal{O}_\Delta)$ such that $[\mathfrak{s}] * j(E_0) = j(E_1)$

1: Decompose $\mathrm{Cl}(\mathcal{O}_\Delta) = \langle [\mathfrak{b}_1] \rangle \oplus \cdots \oplus \langle [\mathfrak{b}_k] \rangle$, where $|\langle [\mathfrak{b}_j] \rangle| = n_j$

2: Solve the hidden shift problem defined by functions $f_0, f_1 \colon \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k} \to \mathrm{Ell}_{q,n}(\mathcal{O}_\Delta)$ satisfying $f_c(x_1, \ldots, x_k) = ([\mathfrak{b}_1]^{x_1} \cdots [\mathfrak{b}_k]^{x_k}) * j(E_c)$, giving some $(s_1, \ldots, s_k) \in \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$

3: Output $[\mathfrak{s}] = [\mathfrak{b}_1]^{s_1} \cdots [\mathfrak{b}_k]^{s_k}$

[1, Theorem 4] applies to non-maximal as well as maximal orders – take $\mathfrak{f}$ in the statement of that theorem to be the conductor of the non-maximal order. By Theorem 4.2 (resp. Theorem 4.3), Step 2 uses $2^{O(\sqrt{\ln|\Delta|})} = L(o(1)) = L(0)$ (resp. $L(\sqrt{2})$) evaluations of the functions f_i. By Corollary 3.3, these functions can be evaluated in time $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$ using Algorithm 2, assuming GRH. Overall, Step 2 takes time

$$L_q(\tfrac{1}{2}, \tfrac{\sqrt{3}}{2} + o(1)) = L_q(\tfrac{1}{2}, \tfrac{\sqrt{3}}{2})$$

using Theorem 4.2, or $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2} + \sqrt{2})$ using Theorem 4.3. The cost of Step 3 is negligible. □

Remark 4.6. Using the improved algorithm for evaluating the complex multiplication action described in Remark 3.4, the running time of Algorithm 3 is improved to $L_q(\frac{1}{2}, \frac{1}{\sqrt{2}} + o(1)) = L_q(\frac{1}{2}, \frac{1}{\sqrt{2}})$ using Theorem 4.2 to solve the hidden shift problem (requiring superpolynomial space), and to $L_q(\frac{1}{2}, \frac{1}{\sqrt{2}} + \sqrt{2}) = L_q(\frac{1}{2}, \frac{3}{\sqrt{2}})$ using Theorem 4.3 (requiring only polynomial space).

Remark 4.7. The running time of the algorithm is ultimately limited by two factors: the best known quantum algorithm for the hidden shift problem runs in superpolynomial time, and the same holds for the best known (classical or quantum) algorithm for computing the complex multiplication action. Improving only one of these results to take polynomial time would still result in a superpolynomial-time algorithm.

5 Subexponential-time, polynomial-space quantum algorithm for the general abelian hidden shift problem

Following Kuperberg's discovery of a subexponential-time quantum algorithm for the hidden shift problem in any finite abelian group A (see [21]), Regev presented

a modification of Kuperberg's algorithm that requires only polynomial space, with a slight increase in the running time [24]. However, Regev only explicitly considered the case $A = \mathbb{Z}_{2^n}$, and while he showed that the running time is $L_{|A|}(\frac{1}{2}, c)$, he did not determine the value of the constant c.

In this section we describe a polynomial-space quantum algorithm for the general abelian hidden shift problem using time $L_{|A|}(\frac{1}{2}, \sqrt{2})$. We assume some familiarity with quantum computation; for general background, see for example [22]. We use several of the same techniques employed by Kuperberg [21, Algorithm 5.1, Theorem 7.1] to go beyond the case $A = \mathbb{Z}_{2^n}$, adapted to work with a Regev-style sieve that only uses polynomial space.

Let $A = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_t}$ be a finite abelian group. Consider the hidden shift problem with hidden shift $s = (s_1, \ldots, s_t) \in A$. By Fourier sampling, one (coherent) evaluation of the hiding functions f_0, f_1 can produce the quantum state

$$|\psi_x\rangle := \frac{1}{\sqrt{2}}\Big(|0\rangle + \exp\Big[2\pi i\Big(\frac{s_1 x_1}{N_1} + \cdots + \frac{s_t x_t}{N_t}\Big)\Big]|1\rangle\Big) \qquad (\psi)$$

with a known label $x = (x_1, \ldots, x_t) \in_{\mathrm{R}} A$, where $x \in_{\mathrm{R}} A$ denotes that x is drawn uniformly at random from A (see for example the proof of [21, Theorem 7.1]). Here $|\psi_x\rangle$ is the state of a single qubit, a vector in the complex vector space $\mathbb{C}^2$ spanned by orthonormal basis states $|0\rangle$ and $|1\rangle$. For simplicity, we begin by considering the case where $A = \mathbb{Z}_N$ is cyclic. Then Fourier sampling produces states

$$|\psi_x\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + \omega^{sx}|1\rangle\big),$$

where $x \in_{\mathrm{R}} \mathbb{Z}_N$ is known and $\omega := e^{2\pi i/N}$.

If we could make states $|\psi_x\rangle$ with chosen values of x, then we could determine s. In particular, the following observation is attributed to Høyer in [21]:

Lemma 5.1. *Given one copy each of the states* $|\psi_1\rangle, |\psi_2\rangle, |\psi_4\rangle, \ldots, |\psi_{2^{k-1}}\rangle$, *where* $2^k = \Omega(N)$, *one can reconstruct* s *in polynomial time with probability* $\Omega(1)$.

Proof. We have

$$\bigotimes_{j=0}^{k-1} |\psi_{2^j}\rangle = \frac{1}{\sqrt{2^k}} \sum_{y=0}^{2^k-1} \omega^{sy}|y\rangle.$$

Apply the inverse quantum Fourier transform over $\mathbb{Z}_N$ (which takes poly$(\log N)$ time [20]) and measure in the computational basis. The Fourier transform of $|s\rangle$,

namely $\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{sy} |s\rangle$, has overlap squared with this state of $2^k/N$, which implies the claim. □

We aim to produce states of the form $|\psi_{2^j}\rangle$ using a sieve that repeatedly combines states to prepare new ones with more desirable labels. The combination procedures used by this sieve all have the following basic structure. First we collect the k input states together, giving a quantum state

$$|\psi_{x_1}\rangle \otimes \cdots \otimes |\psi_{x_k}\rangle = \frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} \omega^{s(x \cdot y)} |y\rangle,$$

where $x \cdot y := \sum_{i=1}^{k} x_i y_i$. Next we compute some function $h(x, y)$ of the known classical value x and the value y in the quantum register. Storing $h(x, y)$ in an ancilla register gives a state of the form

$$\frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} \omega^{s(x \cdot y)} |y\rangle |h(x, y)\rangle.$$

We then measure the ancilla register, giving a superposition over values in the first register that are consistent with the measurement outcome. Finally, we make another measurement to project this superposition onto a particular pair of terms, and relabel these terms as $|0\rangle$ and $|1\rangle$ to produce a state of the form $|\psi_{x'}\rangle$ for some $x' \in \mathbb{Z}_N$.

More concretely, the first such combination procedure is Algorithm 4, which selects pairs of terms with a common quotient in order to produce states with smaller labels.

Lemma 5.2. *Algorithm* 4 *runs in time* $2^k \operatorname{poly}(\log N)$ *and succeeds with probability* $\Omega(1)$ *provided* $4k \leq B/B' \leq 2^k/k$.

Proof. The running time is dominated by the brute force calculation in Step 6 and the projection in Step 10, both of which can be performed in time $2^k \operatorname{poly}(\log N)$.

The probability of aborting in Step 2 for any one x_i is $1 - \frac{2B'}{B} \lfloor \frac{B}{2B'} \rfloor \leq \frac{2B'}{B}$, so by the union bound, the overall probability of aborting in this step is at most $k \frac{2B'}{B} \leq 1/2$. Conditioned on not aborting in Step 2,

$$x_i \in_{\mathrm{R}} \{0, 1, \ldots, 2B' \lfloor B/2B' \rfloor - 1\}.$$

Let $x \cdot y^j = q(2B') + r^j$, where $0 \leq r^j < 2B'$ (q is the measurement outcome, which is independent of j). By the uniformity of the x_is, each $r^j = x \cdot y^j \bmod 2B'$ is uniformly distributed over $\{0, 1, \ldots, 2B' - 1\}$. Thus the output

Algorithm 4 Combining states to give smaller labels

Input: Parameters B, B' and states $|\psi_{x_1}\rangle, \ldots, |\psi_{x_k}\rangle$ with known $x_1, \ldots, x_k \in_{\mathrm{R}} \{0, 1, \ldots, B-1\}$

Output: State $|\psi_{x'}\rangle$ with known $x' \in_{\mathrm{R}} \{0, 1, \ldots, B'-1\}$

1: **if** $\exists i : x_i \geq 2B' \lfloor B/2B' \rfloor$ **then**
2: Abort
3: **end if**
4: Introduce an ancilla and compute

$$\frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} \omega^{s(x \cdot y)} |y\rangle |\lfloor (x \cdot y)/2B' \rfloor\rangle$$

5: Measure the ancilla, giving an outcome q and a state

$$\frac{1}{\sqrt{\nu}} \sum_{j=1}^{\nu} \omega^{s(x \cdot y^j)} |y^j\rangle,$$

where $y^1, \ldots, y^\nu \neq 0^k$ are the k-bit strings such that $\lfloor (x \cdot y^j)/2B' \rfloor = q$
6: Compute $y^1, \ldots, y^\nu$ by brute force
7: **if** $\nu = 1$ **then**
8: Abort
9: **end if**
10: Project onto $\mathrm{span}\{|y^1\rangle, |y^2\rangle\}$ or $\mathrm{span}\{|y^3\rangle, |y^4\rangle\}$ or $\mathrm{span}\{|y^5\rangle, |y^6\rangle\}$ … or $\mathrm{span}\{|y^{2\lfloor \nu/2 \rfloor - 1}\rangle, |y^{2\lfloor \nu/2 \rfloor}\rangle\}$, giving an outcome $\mathrm{span}\{|y^\star\rangle, |y^\blacklozenge\rangle\}$
11: Let $x' = x \cdot (y^\blacklozenge - y^\star)$, where $x \cdot y^\blacklozenge \geq x \cdot y^\star$ without loss of generality
12: **if** $x' \in \{1, \ldots, B'-1\}$ **then**
13: Abort with probability $B'/(2B' - x')$
14: **else if** $x' \in \{B', \ldots, 2B'-1\}$ **then**
15: Abort
16: **end if**
17: Relabel $|y^\star\rangle \mapsto |0\rangle$ and $|y^\blacklozenge\rangle \mapsto |1\rangle$, giving a state $|\psi_{x'}\rangle$

label is $x' = x \cdot (y^\blacklozenge - y^\star) = |r^\blacklozenge - r^\star|$, where $r^\star, r^\blacklozenge \in_{\mathrm{R}} \{0, 1, \ldots, 2B'-1\}$. A simple calculation shows that the distribution of $|r^\blacklozenge - r^\star|$ is

$$\Pr(|r^\blacklozenge - r^\star| = \delta) = \begin{cases} \frac{1}{2B'} & \text{for } \delta = 0, \\ \frac{2B'-\delta}{2B'^2} & \text{for } \delta \in \{1, \ldots, 2B'-1\}. \end{cases}$$

Thus the probability that we abort in Steps 12–16 is $1/2$, and conditioned on not

Algorithm 5 Sieving quantum states

Input: Procedures to prepare states from a set S_0 and to combine k states from S_{i-1} to make a state from S_i with probability at least p for each $i \in \{1, \ldots, m\}$

Output: State from S_m

1: **repeat**
2: **while** for all i we have fewer than k states from S_i **do**
3: Make a state from S_0
4: **end while**
5: Combine k states from some S_i to make a state from S_{i+1} with probability at least p
6: **until** there is a state from S_m

aborting in these steps, $x' \in_{\mathrm{R}} \{0, 1, \ldots, B'-1\}$. Therefore, the algorithm is correct if it reaches Step 17.

It remains to show that the algorithm succeeds with constant probability. We have already bounded the probability that we abort in Step 2 and Steps 12–16. Since $y = 0$ occurs with probability 2^{-k} and at most one state $|y^\nu\rangle$ can be unpaired (and this only happens when ν is odd), the projection in Step 10 fails with probability at most $\nu^{-1} + 2^{-k} \leq 1/3 + o(1)$. We claim that the probability of aborting in Step 8 (i.e., the probability that $\nu = 1$) is also bounded away from 1. Call a value of q bad if $\nu = 1$. Since $0 \leq x \cdot y \leq k(B-1)$, it follows that there are at most $kB/2B'$ possible values of q, and in particular, there can be at most $kB/2B'$ bad values of q. Since the probability of any particular bad q is $1/2^k$, the probability that q is bad is at most $kB/B'2^{k+1} \leq 1/2$. This completes the proof. □

We now apply this combination procedure using the generalized sieve of Algorithm 5, which is equivalent to Regev's "pipeline of routines" [24].

Lemma 5.3. *Suppose* $me^{-2k} = o(1)$. *Then Algorithm* 5 *is correct, succeeds with probability* $1 - o(1)$ *using* $k^{(1+o(1))m}$ *state preparations and combination operations, and uses space* $O(mk)$.

Proof. If Algorithm 5 outputs a state from S_m then it is correct. Since the algorithm never stores more than $O(mk)$ states at a time, it uses space $O(mk)$. It remains to show that the algorithm is likely to succeed using only $k^{(1+o(1))m}$ state preparations and combination operations.

If we could perform combinations deterministically, we would need

$$\begin{aligned} &1 \text{ state from } S_m, \\ &k \text{ states from } S_{m-1}, \\ &k^2 \text{ states from } S_{m-2}, \\ &\vdots \\ &k^m \text{ states from } S_0. \end{aligned}$$

Since the combinations only succeed with probability p, we lower bound the probability of eventually producing $(2k/p)^{m-i}$ states from S_i for each $i \in \{1, \ldots, m\}$ (so in particular, we produce one state from S_m). Given $(2k/p)^{m-i+1}$ states from S_{i-1}, the expected number of successful combinations is $p(2k/p)^{m-i+1}/k = 2(2k/p)^{m-i}$, whereas only $(2k/p)^{m-i}$ successful combinations are needed. By the Chernoff bound, the probability of having fewer than $(2k/p)^{m-i}$ successful combinations is at most $e^{-p(2k/p)^{m-i}}$. Thus, by the union bound, the probability that the algorithm fails is at most

$$\sum_{i=1}^{m-1} e^{-p(2k/p)^{m-i}} \leq m e^{-2k},$$

so the probability of success is $1 - o(1)$.

Finally, the number of states from S_0 is $(2k/p)^m = k^{(1+o(1))m}$ and the total number of combinations is $\sum_{i=0}^{m-1} (2k/p)^{m-i}/k = k^{(1+o(1))m}$. □

When using the sieve, we have the freedom to choose the relationship between k and m to optimize the running time. Suppose that $mk = (1 + o(1)) \log_2 N$ (intuitively, to cancel $\log_2 N$ bits of the label), and also suppose that the combination operation takes time $2^k \operatorname{poly}(\log N)$ (as in Lemma 5.2). Then if we take $k = c\sqrt{\log_2 N \log_2 \log_2 N}$, we find that the overall running time of Algorithm 5 is

$$2^k 2^{(1+o(1))m \log_2 k} \operatorname{poly}(\log N) = L_N(\tfrac{1}{2}, c + \tfrac{1}{2c}).$$

Choosing $c = \frac{1}{\sqrt{2}}$ gives the best running time, $L_N(\frac{1}{2}, \sqrt{2})$.

We now consider how to apply the sieve. To use Lemma 5.1, our goal is to prepare states of the form $|\psi_{2^j}\rangle$ for each $j \in \{0, 1, \ldots, \lfloor \log_2 N \rfloor\}$. First we show how to prepare the state $|\psi_1\rangle$ in time $L_N(\frac{1}{2}, \sqrt{2})$ using Algorithm 4 as the combination procedure in Algorithm 5. For $i \in \{0, 1, \ldots, m\}$, the ith stage of the sieve produces states with labels from $S_i = \{0, 1, \ldots, B_i - 1\}$. Lemma 5.4 below shows that there is a choice of the B_is with $B_0 = N$, $B_m = 2$, and successive

ratios of the B_is satisfying the conditions of Lemma 5.2, such that $2^k k^{(1+o(1))m} = L_N(\frac{1}{2}, \sqrt{2})$. It then follows that Algorithm 5 produces a uniformly random label from $S_m = \{0,1\}$ with constant probability in time $L_N(\frac{1}{2}, \sqrt{2})$, and in particular, can be used to produce a copy of $|\psi_1\rangle$ in time $L_N(\frac{1}{2}, \sqrt{2})$.

Lemma 5.4. *There is a constant N_0 such that for all $N > N_0$, letting $B_i = \lfloor N/\rho^i \rfloor$, where $\rho = (N/2)^{1/m}$ and*

$$k = \left\lfloor \sqrt{\tfrac{1}{2} \log_2 N \log_2 \log_2 N} \right\rfloor, \quad m = \left\lceil \frac{\log_2 N/2}{k - \log_2 2k} \right\rceil = \Theta\Big(\sqrt{\frac{\log_2 N}{\log_2 \log_2 N}} \Big),$$

we have $B_0 = N$, $B_m = 2$, and $4k \leq B_{i-1}/B_i \leq 2^k/k$ for all $i \in \{1, \ldots, m\}$.

Proof. Clearly $B_0 = N$, and the value of ρ is chosen so that $B_m = 2$.

For $i \in \{1, \ldots, m\}$, we have

$$\frac{B_{i-1}}{B_i} = \frac{\lfloor N/\rho^{i-1} \rfloor}{\lfloor N/\rho^i \rfloor} \leq \frac{N/\rho^{i-1}}{N/\rho^i - 1} = \frac{\rho}{1 - \rho^i/N}.$$

Since $\rho^i/N \leq \rho^m/N = 1/2$, we have $B_{i-1}/B_i \leq 2\rho$. Then using

$$\rho \leq (N/2)^{\frac{k - \log_2 2k}{\log_2 N/2}} = \frac{2^k}{2k}$$

gives $B_{i-1}/B_i \leq 2^k/k$ as claimed. Similarly, we have

$$\frac{B_{i-1}}{B_i} = \frac{\lfloor N/\rho^{i-1} \rfloor}{\lfloor N/\rho^i \rfloor} \geq \frac{N/\rho^{i-1} - 1}{N/\rho^i} = \rho - \frac{\rho^i}{N} \geq \rho - \frac{1}{2}.$$

Since

$$\rho = (N/2)^{\Theta(\sqrt{\log_2 \log_2 N / \log_2 N})} = 2^{\Theta(\sqrt{\log_2 N \log_2 \log_2 N})} = 2^{\Theta(k)},$$

we have $\rho - \frac{1}{2} \geq 4k$ for sufficiently large N. This completes the proof. □

If N is odd, then division by 2 is an automorphism of $\mathbb{Z}_N$. Thus we can prepare $|\psi_{2^j}\rangle$ by performing the above sieve under the automorphism $x \mapsto 2^{-j}x$. It follows that the abelian hidden shift problem in a cyclic group of odd order N can be solved in time $L_N(\frac{1}{2}, \sqrt{2})$.

Now suppose that $N = 2^n$ is a power of 2. In this case, we first use a combination procedure that zeros out low-order bits, as described in Algorithm 6. We use the notation $xS := \{xz : z \in S\}$ for any $x \in \mathbb{Z}$ and $S \subset \mathbb{Z}$.

Algorithm 6 Combining states to cancel low-order bits

Input: Parameters ℓ, ℓ' and states $|\psi_{x_1}\rangle, \ldots, |\psi_{x_k}\rangle$ with known $x_1, \ldots, x_k \in_{\mathrm{R}} 2^{\ell}\{0, 1, \ldots, N/2^{\ell} - 1\}$

Output: State $|\psi_{x'}\rangle$ with known $x' \in_{\mathrm{R}} 2^{\ell'}\{0, 1, \ldots, N/2^{\ell'} - 1\}$

1: Introduce an ancilla and compute

$$\frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} \omega^{s(x \cdot y)} |y\rangle |x \cdot y \bmod 2^{\ell'}\rangle$$

2: Measure the ancilla, giving an outcome r and a state

$$\frac{1}{\sqrt{\nu}} \sum_{j=1}^{\nu} \omega^{s(x \cdot y^j)} |y^j\rangle,$$

where $y^1, \ldots, y^{\nu} \neq 0^k$ are the k-bit strings such that $x \cdot y^j \bmod 2^{\ell'} = r$

3: Compute $y^1, \ldots, y^{\nu}$ by brute force

4: **if** $\nu = 1$ **then**

5: Abort

6: **end if**

7: Project onto $\mathrm{span}\{|y^1\rangle, |y^2\rangle\}$ or $\mathrm{span}\{|y^3\rangle, |y^4\rangle\}$ or $\mathrm{span}\{|y^5\rangle, |y^6\rangle\}$ or $\ldots$ or $\mathrm{span}\{|y^{2\lfloor \nu/2 \rfloor - 1}\rangle, |y^{2\lfloor \nu/2 \rfloor}\rangle\}$, giving an outcome $\mathrm{span}\{|y^{\star}\rangle, |y^{\blacklozenge}\rangle\}$

8: Relabel $|y^{\star}\rangle \mapsto |0\rangle$ and $|y^{\blacklozenge}\rangle \mapsto |1\rangle$, giving a state $|\psi_{x'}\rangle$ with $x' = x \cdot (y^{\blacklozenge} - y^{\star}) \bmod N$

Lemma 5.5. *Algorithm* 6 *runs in time* $2^k \operatorname{poly}(\log N)$ *and succeeds with probability* $\Omega(1)$ *provided* $k \geq \ell' - \ell + 1$.

Proof. The proof is similar to that of Lemma 5.2. Again the running time is dominated by the brute force calculation in Step 3 and the projection in Step 7, both of which can be performed in time $2^k \operatorname{poly}(\log N)$.

We claim that the algorithm is correct if it reaches Step 8. Observe that $x \cdot y^j \bmod N = q^j 2^{\ell'} + r$, where r is independent of j. Since $y^j \neq 0^k$, we have $x \cdot y^j \bmod N \in_{\mathrm{R}} 2^{\ell}\{0, 1, \ldots, N/2^{\ell} - 1\}$, so $q^j \in_{\mathrm{R}} \{0, 1, \ldots, N/2^{\ell'} - 1\}$, and hence

$$x' = (q^{\blacklozenge} - q^{\star}) 2^{\ell'} \bmod N \in_{\mathrm{R}} 2^{\ell'}\{0, 1, \ldots, N/2^{\ell'} - 1\}$$

as required.

The projection in Step 7 fails with probability at most $1/3 + o(1)$. It remains to show that the algorithm reaches Step 7 with probability $\Omega(1)$, i.e., to upper bound

the probability that $\nu = 1$. Call a value of r bad if $\nu = 1$. There are $2^{\ell'-\ell}$ possible values of r, so in particular there are at most $2^{\ell'-\ell}$ bad values of r. Since the probability of any particular bad r is $1/2^k$, the probability that r is bad is at most $2^{\ell'-\ell-k} \leq 1/2$. This completes the proof. □

Algorithm 6 is similar to the combination procedure used in [24], but differs in that the latter requires $\nu = O(1)$, which is established in the analysis using a second moment argument. The modification of pairing as many values of y as possible allows us to use a simpler analysis (with essentially the same performance).

To produce a state of the form $|\psi_{2^j}\rangle$, we first use Algorithm 6 to cancel low-order bits and then use Algorithm 4 to cancel high-order bits. Note that if all states $|\psi_x\rangle$ have labels x with a common factor – say, $2^j \,|\, x$ – then we can view the labels as elements of $\mathbb{Z}_{2^{n-j}}$ and apply Algorithm 4 to affect the $n-j$ most significant bits. Specifically, to make the state $|\psi_{2^j}\rangle$, we apply Algorithm 5 using Algorithm 6 as the combination procedure that produces states from S_i using states from S_{i-1} for $i \in \{1, \ldots, m_1 + 1\}$, and Algorithm 4 (on the $n-j$ most significant bits) as the combination procedure for $i \in \{m_1 + 2, \ldots, m_1 + m_2 + 1\}$, taking

$$S_i = \begin{cases} 2^{(k-1)i}\{0, 1, \ldots, 2^{n-(k-1)i} - 1\} & \text{for } i \in \{0, 1, \ldots, m_1\}, \\ 2^j\{0, 1, \ldots, B_i - 1\} & \text{for } i \in \{m_1 + 1, \ldots, m_1 + m_2 + 1\}, \end{cases}$$

where now

$$B_i = \left\lfloor \frac{2^{n-j}}{\rho^{i-m_1-1}} \right\rfloor, \qquad m_1 = \left\lfloor \frac{j}{k-1} \right\rfloor,$$
$$\rho = 2^{(n-j-1)/m_2}, \qquad m_2 = \left\lceil \frac{n-j}{k - \log_2 2k} \right\rceil,$$

and again

$$k = \left\lfloor \sqrt{\tfrac{1}{2} \log_2 N \log_2 \log_2 N} \right\rfloor.$$

When making states in S_i from states in S_{i-1} for $i \in \{1, \ldots, m_1\}$, we cancel $k-1$ bits with k states, so the condition of Lemma 5.5 is satisfied. For $i = m_1 + 1$, we cancel

$$j - (k-1)m_1 = j - (k-1)\left\lfloor \frac{j}{k-1} \right\rfloor \leq j - (k-1)\left\lceil \frac{j}{k-1} - 1 \right\rceil = k - 1$$

bits, so again the condition of Lemma 5.5 is satisfied. For $i \in \{m_1 + 2, \ldots, m_1 + m_2 + 1\}$, Lemma 5.4 implies that the conditions of Lemma 5.2 are satisfied provided $2^{n-j} \geq N_0$. (If $2^{n-j} < N_0$ then we only need to perform the first $m_1 + 1$ stages of the sieve, producing a state uniformly at random from S_{m_1+1}; in this case

$|S_{m_1+1}| = O(1)$, so $O(1)$ repetitions suffice to produce a copy of $|\psi_{2^j}\rangle$.) Finally, since $(m_1 + m_2 + 1)k = (1 + o(1))n$, the discussion following Lemma 5.3 shows that Algorithm 5 takes time $L_N(\frac{1}{2}, \sqrt{2})$.

So far we have covered the case where the group is $A = \mathbb{Z}_N$ with N either odd or a power of 2. Now consider the case of a general finite abelian group $A = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_t}$. By the Chinese remainder theorem, we can assume without loss of generality that each N_i is either odd or a power of 2. Consider what happens if we apply Algorithm 4 or Algorithm 6 to one component of a product of cyclic groups. Suppose we combine k states of the form of equation (ψ). For each $i \in \{1, \ldots, k\}$, let $x_i \in \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_t}$ denote the label of the ith state, with $x_{i,j} \in \mathbb{Z}_{N_j}$ for $j \in \{1, \ldots, t\}$. To address the ℓth component of A, the combination procedure prepares a state

$$\frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} \exp\Bigl(2\pi i \sum_{i=1}^{k} \sum_{j=1}^{t} \frac{y_i x_{i,j} s_j}{N_j}\Bigr) |y\rangle |h(\textstyle\sum_{i=1}^{k} x_{i,\ell} y_i)\rangle$$

for some function h (a quotient in Algorithm 4 or a remainder in Algorithm 6). For $j \neq \ell$, if $x_{i,j} = 0$ for all $i \in \{1, \ldots, k\}$ then $x'_j = \sum_{i=1}^{k} x_{i,j}(y_i^{\blacklozenge} - y_i^{\star}) = 0$, so components that are initially zero remain zero. Thus, if we can prepare states $|\psi_x\rangle$ with $x_\ell \in_{\mathrm{R}} \mathbb{Z}_{N_\ell}$ (for any desired $\ell \in \{1, \ldots, t\}$) and all other components zero, we effectively reduce the problem to the cyclic case.

To prepare such states, we use a new combination procedure, Algorithm 7. Without loss of generality, our goal is to zero out the first $t - 1$ components, leaving the last one uniformly random from $\mathbb{Z}_{N_t}$. Algorithm 7 is similar to Algorithm 4, viewing the first $t - 1$ components of the label $x_i \in \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_t}$ as the mixed-radix integer

$$\mu(x_i) := \sum_{j=1}^{t-1} x_{i,j} \prod_{j'=1}^{j-1} N_{j'}.$$

Because we are merely trying to zero out certain components, we no longer require uniformity of the states output by the sieve, which simplifies the procedure and its analysis.

Lemma 5.6. *Algorithm* 7 *runs in time* $2^k \operatorname{poly}(\log N)$ *and succeeds with probability* $\Omega(1)$ *provided* $B/B' \leq 2^k/2k$.

Proof. As in Lemma 5.2 and Lemma 5.5, the running time is dominated by the brute force calculation in Step 3 and the projection in Step 7, both of which can be performed in time $2^k \operatorname{poly}(\log N)$.

Algorithm 7 Combining non-cyclic states to reduce undesired components

Input: Parameters B, B' and states $|\psi_{x_1}\rangle, \ldots, |\psi_{x_k}\rangle$ with known $x_1, \ldots, x_k \in \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_t}$ satisfying $\mu(x_i) \in \{0, 1, \ldots, B-1\}$ for each $i \in \{1, \ldots, k\}$, with $x_{i,t} \in_{\mathrm{R}} \mathbb{Z}_{N_t}$

Output: State $|\psi_{x'}\rangle$ with known $x' \in \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_t}$ satisfying $\mu(x') \in \{0, 1, \ldots, B'-1\}$, with $x'_t \in_{\mathrm{R}} \mathbb{Z}_{N_t}$

1: Introduce an ancilla register and compute

$$\frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} \exp\Big(2\pi i \sum_{i=1}^{k} \sum_{j=1}^{t} \frac{y_i x_{i,j} s_j}{N_j}\Big) |y\rangle |\lfloor \textstyle\sum_{i=1}^{k} \mu(x_i) y_i / B' \rfloor\rangle$$

2: Measure the ancilla register, giving an outcome q and a state

$$\frac{1}{\sqrt{\nu}} \sum_{j=1}^{\nu} \exp\Big(2\pi i \sum_{i=1}^{k} \sum_{j=1}^{t} \frac{y_i x_{i,j} s_j}{N_j}\Big) |y^j\rangle,$$

where $y^1, \ldots, y^\nu \neq 0^k$ are the k-bit strings such that $\lfloor \sum_{i=1}^{k} \mu(x_i) y_i^j / B' \rfloor = q$

3: Compute $y^1, \ldots, y^\nu$ by brute force

4: **if** $\nu = 1$ **then**

5: Abort

6: **end if**

7: Project onto $\mathrm{span}\{|y^1\rangle, |y^2\rangle\}$ or … or $\mathrm{span}\{|y^{2\lfloor \nu/2 \rfloor - 1}\rangle, |y^{2\lfloor \nu/2 \rfloor}\rangle\}$, giving an outcome $\mathrm{span}\{|y^\star\rangle, |y^\blacklozenge\rangle\}$

8: Relabel $|y^\star\rangle \mapsto |0\rangle$ and $|y^\blacklozenge\rangle \mapsto |1\rangle$, where $\sum_{i=1}^{k} \mu(x_i) y_i^\blacklozenge \geq \sum_{i=1}^{k} \mu(x_i) y_i^\star$ without loss of generality, giving a state $|\psi_{x'}\rangle$ with $x'_j = \sum_{i=1}^{k} x_{i,j} (y_i^\blacklozenge - y_i^\star)$ for each $j \in \{1, \ldots, t\}$

We claim that the algorithm is correct if it reaches Step 8. In this case, since $\sum_{i=1}^{k} \mu(x_i) y_i^j = qB' + r^j$, where q is independent of j and $0 \leq r^j < B'$, we have

$$\begin{aligned} \mu(x') &= \sum_{j=1}^{t-1} x'_j \prod_{j'=1}^{j-1} N_{j'} = \sum_{j=1}^{t-1} \sum_{i=1}^{k} x_{i,j} (y_i^\blacklozenge - y_i^\star) \prod_{j'=1}^{j-1} N_{j'} \\ &= \sum_{i=1}^{k} \mu(x_i)(y_i^\blacklozenge - y_i^\star) = r^\blacklozenge - r^\star < B' \end{aligned}$$

as required. Since $y^\star$ differs from $y^\blacklozenge$ and the $x_{i,t}$ are uniformly random, $x'_t = \sum_{i=1}^{k} x_{i,t}(y_i^\blacklozenge - y_i^\star)$ is uniformly random as required.

The projection in Step 7 fails with probability at most $1/3 + o(1)$. We claim the algorithm reaches Step 7 with probability $\Omega(1)$. To show this, we need to upper bound the probability that $\nu = 1$. Call a value of q bad if $\nu = 1$. Since $0 \le \sum_{i=1}^{k} \mu(x_i) y_i \le k(B-1)$, there are at most kB/B' possible values of q, and in particular, there can be at most kB/B' bad values of q. Since the probability of any particular bad q is $1/2^k$, the probability that q is bad is at most $kB/B'2^k \le 1/2$. This completes the proof. □

To apply Algorithm 7 as the combination procedure for Algorithm 5, we require a straightforward variant of Lemma 5.4, as follows.

Lemma 5.7. *There is a constant N'_0 such that for all $N > N'_0$, letting $B_i = \lfloor N/\rho^i \rfloor$, where $\rho = N^{1/m}$ and*

$$k = \left\lfloor \sqrt{\tfrac{1}{2} \log_2 N \log_2 \log_2 N} \right\rfloor \quad m = \left\lceil \frac{\log_2 N}{k - \log_2 4k} \right\rceil = \Theta\Big(\sqrt{\frac{\log_2 N}{\log_2 \log_2 N}} \Big),$$

we have $B_0 = N$, $B_m = 1$, and $B_{i-1}/B_i \le 2^k/2k$ for all $i \in \{1, \ldots, m\}$.

Proof. Clearly $B_0 = N$, and the value of ρ is chosen so that $B_m = 1$.

We have $B_{m-1} \le N/\rho^{m-1} = \rho$, and since

$$\rho \le N^{\frac{k - \log_2 4k}{\log_2 N}} = \frac{2^k}{4k},$$

the claimed inequality holds for $i = m$. For $i \in \{1, \ldots, m-1\}$,

$$\frac{B_{i-1}}{B_i} = \frac{\lfloor N/\rho^{i-1} \rfloor}{\lfloor N/\rho^i \rfloor} \le \frac{N/\rho^{i-1}}{N/\rho^i - 1} = \frac{\rho}{1 - \rho^i/N}.$$

Since $\rho^i/N \le \rho^{m-1}/N = 1/\rho$, we have $B_{i-1}/B_i \le \rho/(1 - 1/\rho)$. Then using

$$\rho = N^{\Theta(\sqrt{\log_2 \log_2 N / \log_2 N})} = 2^{\Theta(\sqrt{\log_2 N \log_2 \log_2 N})},$$

we have $\rho \ge 2$ provided $N > N'_0$ for some constant N'_0, which implies $B_{i-1}/B_i \le 2^k/2k$. This completes the proof. □

Combining these ideas, the overall procedure is presented in Algorithm 8.

Theorem 5.8. *Algorithm* 8 *runs in time $L_{|A|}(\frac{1}{2}, \sqrt{2})$.*

Algorithm 8 Abelian hidden shift problem

Input: Black box for the hidden shift problem in an abelian group A
Output: Hidden shift s

1: Write $A = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_t}$, where each N_i is either odd or a power of 2
2: **for all** $i \in \{1, \ldots, t\}$ **do**
3: **if** N_i is odd **then**
4: **for all** $j \in \{0, \ldots, \lfloor \log_2 N_i \rfloor\}$ **do**
5: Apply Algorithm 5, first using Algorithm 7 (as the subroutine in Step 5) to zero out all components except the ith, and then using Algorithm 4 under the $\mathbb{Z}_{N_i}$-automorphism $x \mapsto 2^{-j}x$ to produce a copy of $|\psi_{(0,\ldots,0,2^j,0,\ldots,0)}\rangle$ (see the proof of Theorem 5.8 for detailed parameters)
6: **end for**
7: **else**
8: Let $N_i = 2^n$
9: **for all** $j \in \{0, \ldots, n-1\}$ **do**
10: Apply Algorithm 5, first using Algorithm 7 to zero out all components except the ith, then using Algorithm 6 to make states $|\psi_{(0,\ldots,0,x,0,\ldots,0)}\rangle$ with $2^j | x$, and finally using Algorithm 4 to produce a copy of $|\psi_{(0,\ldots,0,2^j,0,\ldots,0)}\rangle$ (see the proof of Theorem 5.8 for detailed parameters)
11: **end for**
12: **end if**
13: Apply Lemma 5.1 with $N = N_i$ to give s_i
14: **end for**
15: Output $s = (s_1, \ldots, s_t)$

Proof. In Step 1, if the structure of the group is not initially known, it can be determined in polynomial time using [8]. Given the structure of the group, for each term $\mathbb{Z}_N$ we can easily factor $N = 2^n M$, where M is odd; then $\mathbb{Z}_N \cong \mathbb{Z}_{2^n} \times \mathbb{Z}_M$, and we obtain a decomposition of the desired form.

Now suppose without loss of generality that we are trying to determine s_t (i.e., $i = t$ in Step 2). The main contribution to the running time comes from the sieves in Step 5 (for N_t odd) and Step 10 (for N_t a power of 2).

First suppose that N_t is odd. It suffices to handle the case where $j = 0$, so we are making the state $|\psi_{(0,\ldots,0,1)}\rangle$. Then we apply Algorithm 5 with

$$S_i = \begin{cases} \{x \in A : \mu(x) < B_i\} & \text{for } i \in \{0, 1, \ldots, m_1\}, \\ \{x \in A : \mu(x) = 0 \text{ and } x_t < B_i\} & \text{for } i \in \{m_1 + 1, \ldots, m_2\}, \end{cases}$$

where

$$B_i = \begin{cases} \lfloor (N/N_t)/\rho_1^i \rfloor & \text{for } i \in \{0, 1, \ldots, m_1\}, \\ \lfloor N_t/\rho_2^{i-m_1} \rfloor & \text{for } i \in \{m_1+1, \ldots, m_1+m_2\} \end{cases}$$

with

$$\rho_1 = (N/N_t)^{1/m_1}, \qquad m_1 = \left\lceil \frac{\log_2 N/N_t}{k - \log_2 4k} \right\rceil,$$

$$\rho_2 = (N_t/2)^{1/m_2}, \qquad m_2 = \left\lceil \frac{\log_2 N_t/2}{k - \log_2 2k} \right\rceil,$$

and

$$k = \left\lfloor \sqrt{\tfrac{1}{2} \log_2 N \log_2 \log_2 N} \right\rfloor.$$

We use Algorithm 7 as the combination procedure for the first m_1 stages of Algorithm 5. By Lemma 5.7, the condition of Lemma 5.6 is satisfied provided $N/N_t > N_0'$; otherwise we can produce a state with a label from S_{m_1} in only $O(1)$ trials. Then we proceed to apply Algorithm 4 as the combination procedure for the remaining m_2 stages of Algorithm 5. By Lemma 5.4, the conditions of Lemma 5.2 are satisfied provided $N_t > N_0$; otherwise, producing states with labels from S_{m_1} already suffices to produce the desired state with constant probability. Since $(m_1 + m_2)k = (1 + o(1)) \log_2 N$, Step 5 takes time $L_{|A|}(\frac{1}{2}, \sqrt{2})$ (see the discussion following the proof of Lemma 5.3).

Now suppose that $N_t = 2^n$ is a power of 2. Then we apply Algorithm 5 with

$$S_i = \begin{cases} \{x \in A : \mu(x) < B_i\} & \text{for } i \in \{0, 1, \ldots, m_1\}, \\ \{x \in A : \mu(x) = 0 \text{ and } \\ \quad x_t \in 2^{(k-1)i}\{0, 1, \ldots, 2^{n-(k-1)i}\}\} & \text{for } i \in \{m_1+1, \ldots, m_1+m_2\}, \\ \{x \in A : \mu(x) = 0 \text{ and } \\ \quad x_t \in 2^j\{0, 1, \ldots, B_i - 1\}\} & \text{for } i \in \{m_1+m_2+1, \ldots, \\ & \qquad m_1+m_2+m_3+1\}, \end{cases}$$

where

$$B_i = \begin{cases} \lfloor (N/N_t)/\rho_1^i \rfloor & \text{for } i \in \{0, 1, \ldots, m_1\}, \\ \lfloor 2^{n-j}/\rho_3^{i-m_1-m_2-1} \rfloor & \text{for } i \in \{m_1+m_2+1, \ldots, \\ & \qquad m_1+m_2+m_3+1\} \end{cases}$$

with

$$\rho_1 = (N/N_t)^{1/m_1}, \qquad m_1 = \left\lceil \frac{\log_2 N/N_t}{k - \log_2 4k} \right\rceil,$$

$$\rho_3 = 2^{(n-j-1)/m_3}, \qquad m_2 = \left\lfloor \frac{j}{k-1} \right\rfloor, \qquad m_3 = \left\lceil \frac{n-j-1}{k - \log_2 2k} \right\rceil,$$

and again

$$k = \left\lfloor \sqrt{\tfrac{1}{2} \log_2 N \log_2 \log_2 N} \right\rfloor.$$

We use Algorithm 7 as the combination procedure for the first m_1 stages, Algorithm 6 for the next $m_2 + 1$ stages, and Algorithm 4 (on the $n - j$ most significant bits) for the final m_3 stages. By Lemma 5.7 and Lemma 5.4, the conditions of Lemma 5.6 and Lemma 5.2 are satisfied, respectively. Since we cancel at most $k - 1$ bits in each stage that uses Algorithm 6, the conditions of Lemma 5.5 are satisfied for the intermediate stages. Finally, since $(m_1 + m_2 + m_3 + 1)k = (1 + o(1)) \log_2 N$, Step 10 takes time $L_{|A|}(\frac{1}{2}, \sqrt{2})$.

The loops in Step 2, Step 4, and Step 9 only introduce polynomial overhead. Step 13 takes polynomial time and Step 15 is negligible. Thus the overall running time is $L_{|A|}(\frac{1}{2}, \sqrt{2})$ as claimed. □

Bibliography

[1] E. Bach, Explicit bounds for primality testing and related problems, *Math. Comp.* **55** (1990), 355–380.

[2] C. H. Bennett, E. Bernstein, G. Brassard and U. Vazirani, Strengths and weaknesses of quantum computing, *SIAM J. Comput.* **26** (1997), 1510–1523.

[3] D. J. Bernstein, How to find smooth parts of integers, preprint (2004), `http://cr.yp.to/papers.html#smoothparts`.

[4] G. Bisson, Computing endomorphism rings of elliptic curves under the GRH, *J. Math. Cryptol.* **5** (2011), 101–113.

[5] G. Bisson and A. V. Sutherland, Computing the endomorphism ring of an ordinary elliptic curve over a finite field, *J. Number Theory* **131** (2011), 815–831.

[6] R. Bröker, D. Charles and K. Lauter, Evaluating large degree isogenies and applications to pairing based cryptography, in: *Pairing '08: Proceedings of the 2nd International Conference on Pairing-Based Cryptography*, Lecture Notes in Comput. Sci. 5209, Springer, Berlin (2008), 100–112.

[7] J. Buchmann and U. Vollmer, *Binary Quadratic Forms: An Algorithmic Approach*, Algorithms Comput. Math. 20, Springer, Berlin, 2007.

[8] K. K. H. Cheung and M. Mosca, Decomposing finite abelian groups, *Quantum Inform. Comput.* **1** (2001), 26–32.

[9] J.-M. Couveignes, Hard homogeneous spaces, preprint (2006), `http://eprint.iacr.org/2006/291`.

[10] D. A. Cox, *Primes of the Form $x^2 + ny^2$: Fermat, Class Field Theory and Complex Multiplication*, Wiley, New York, 1989.

[11] W. van Dam, S. Hallgren and L. Ip, Quantum algorithms for some hidden shift problems, in: *SODA '02: Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, (2002), 489–498.

[12] M. Ettinger and P. Høyer, On quantum algorithms for noncommutative hidden subgroups, *Adv. Appl. Math.* **25** (2000), 239–251.

[13] S. D. Galbraith, F. Hess and N. P. Smart, Extending the GHS Weil descent attack, in: *Advances in Cryptology* (EUROCRYPT 2002), Lecture Notes in Comput. Sci. 2332, Springer, Berlin (2002), 29–44.

[14] S. D. Galbraith and A. Stolbunov, Improved algorithm for the isogeny problem for ordinary elliptic curves, preprint (2011), `http://arxiv.org/abs/1105.6331v1`.

[15] S. Hallgren, Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem, *J. ACM* **54** (2007), article 4, preliminary version in STOC '02.

[16] J. Hermans, F. Vercauteren and B. Preneel, Speed records for NTRU, in: *CT-RSA*, Lecture Notes in Comput. Sci. 5985, Springer, Berlin (2010), 73–88.

[17] D. Jao and L. De Feo, Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies, in: *PQCrypto*, Lecture Notes in Comput. Sci. 7071, Springer, Berlin (2011), 19–34.

[18] D. Jao, S. D. Miller and R. Venkatesan, Expander graphs based on GRH with an application to elliptic curve cryptography, *J. Number Theory* **129** (2009), 1491–1504.

[19] D. Jao and V. Soukharev, A subexponential algorithm for evaluating large degree isogenies, in: *Algorithmic number theory: Proceedings of ANTS-IX*, Lecture Notes in Comput. Sci. 6197, Springer, Berlin (2010), 219–233.

[20] A. Y. Kitaev, Quantum measurements and the abelian stabilizer problem, preprint (1995), `http://arxiv.org/abs/quant-ph/9511026v1`.

[21] G. Kuperberg, A subexponential-time quantum algorithm for the dihedral hidden subgroup problem, *SIAM J. Comput.* **35** (2005), 170–188.

[22] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, 2000.

[23] R. A. Perlner and D. A. Cooper, Quantum resistant public key cryptography: A survey, in: *Proceedings of the 8th Symposium on Identity and Trust on the Internet* (IDtrust '09), ACM, New York (2009), 85–93.

[24] O. Regev, A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space, preprint (2004), http://arxiv.org/abs/quant-ph/0406151v1.

[25] A. Rostovtsev and A. Stolbunov, Public-key cryptosystem based on isogenies, preprint (2006), http://eprint.iacr.org/2006/145.

[26] R. Schoof, Counting points on elliptic curves over finite fields, *J. Théor. Nombres Bordeaux* **7** (1995), 219–254.

[27] M. Seysen, A probabilistic factorization algorithm with quadratic forms of negative discriminant, *Math. Comp.* **48** (1987), 757–780.

[28] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26** (1997), 1484–1509, preliminary version in FOCS '94.

[29] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Math. 106, Springer, New York, 1992. Corrected reprint of the 1986 original.

[30] A. Stolbunov, Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves, *Adv. Math. Commun.* **4** (2010), 215–235.

[31] J. Tate, Endomorphisms of abelian varieties over finite fields, *Invent. Math.* **2** (1966), 134–144.

[32] W. C. Waterhouse, Abelian varieties over finite fields, *Ann. Sci. École Norm. Sup. (4)* **2** (1969), 521–560.

Received June 29, 2012; revised June 7, 2013; accepted September 29, 2013.

Author information

Andrew Childs, Department of Combinatorics & Optimization and Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.
E-mail: amchilds@math.uwaterloo.ca

David Jao, Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.
E-mail: djao@math.uwaterloo.ca

Vladimir Soukharev, Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.
E-mail: vsoukhar@math.uwaterloo.ca