Another Look at the Quantum Security of the Vectorization Problem with Shifted Inputs

Paul Frixons[1], Valerie Gilchrist[1], Péter Kutas[2,3], Simon-Philipp Merz[4], Christophe Petit[1,3]

[1] Université Libre de Bruxelles, Belgium
[2] Eötvös Loránd University, Hungary
[3] University of Birmingham, United Kingdom
[4] ETH Zürich, Switzerland

Abstract. Cryptographic group actions provide simple post-quantum generalizations to many cryptographic protocols based on the discrete logarithm problem (DLP). However, many advanced group action-based protocols do not solely rely on the core group action problem (the so-called vectorization problem), but also on *variants* of this problem, to either improve efficiency or enable new functionalities. In particular, the security of the CSI-SharK threshold signature protocol relies on the *Vectorization Problem with Shifted Inputs* where (in DLP formalism) the adversary not only receives g and g^x, but also g^{x^c} for multiple known values of c. A natural open question is then whether the extra data provided to the adversary in this variant allows for more efficient attacks.

In this paper, we revisit the concrete quantum security of this problem. We start from a quantum *multiple* hidden shift algorithm of Childs and van Dam, which to the best of our knowledge was never applied in cryptography before. We specify algorithms for its subroutines and we provide concrete complexity estimates for both these subroutines and the overall algorithm.

We then apply our analysis to the CSI-SharK protocol. In prior analyses based on Kuperberg's algorithms, group action evaluations contributed to a significant part of the overall T-gate cost. For CSI-Shark suggested parameters, our new approach requires significantly fewer calls to the group action evaluation subroutine, leading to significant T-gate complexity improvements overall. We also show that the quantum security

* Authors listed in alphabetical order: see https://www.ams.org/profession/leaders/CultureStatement04.pdf. Valerie Gilchrist is supported by a FRIA grant by the National Fund for Scientific Research (F.N.R.S.) of Belgium; Simon-Philipp Merz is supported by DFG through a Walter Benjamin Fellowship and the Zurich Information Security and Privacy Center (ZISC); Christophe Petit and Péter Kutas are partly supported by EPSRC through grant number EP/V011324/1. Péter Kutas was supported by the Ministry of Culture and Innovation and the National Research, Development, and Innovation Office within the Quantum Information National Laboratory of Hungary (Grant No. 2022-2.1.1-NL-2022-00004) and by the grant "EXCELLENCE-151343". Péter Kutas is also supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences

Date of this document: 2025-02-27.

of the protocol decreases when the number of public keys increases, and quantify this degradation.
Beyond its direct application to the CSI-SharK protocol, our work more generally questions the quantum security of vectorization problem variants, and it introduces the Childs-van Dam algorithm as a new quantum cryptanalysis tool.

1 Introduction

The shift towards post-quantum cryptography is extremely important due to the looming quantum threat. This motivates the development of new forms of cryptography based on alternative "hard problems" with the potential to resist quantum computers. Following a massive investigation effort from the cryptography community, we now have good candidates for basic protocols such as signatures and key encapsulation mechanisms. However, realizing advanced protocols based on many of the new problems remains a daunting task.

Cryptographic group actions [5] provide a natural generalization to the classical discrete logarithm problem (DLP), namely the *vectorization problem*, which in turns leads to relatively easy adaptations of many DLP-based protocols.

The most promising post-quantum group action instantiations currently come from isogenies. In particular, CSIDH [24] is a key exchange scheme that replaces classical DLP in the Diffie-Hellman protocol with an isogeny group action. It has led to several other protocols that aim to address different cryptographic needs including signatures [12], ring signatures [40], threshold signatures [7,34], identification protocols [8], and many more. Isogeny-based protocols also benefit from relatively small key sizes, as well as theoretical and practical know-how originating from related elliptic curve cryptography.

If isogeny-based protocols are to be deployed as post-quantum replacements of current DLP-based protocols, they must of course withstand quantum cryptanalysis, and we must understand their exact quantum security. The main quantum attacks on CSIDH and variants so far are Kuperberg's algorithms and variants [11, 21, 45, 46, 53]. In particular, the latest concrete estimates for CSIDH parameters were provided by Peikert at EUROCRYPT 2021 [53].

While our understanding of the quantum security of the vectorization problem for CSIDH has greatly improved recently, several advanced protocols based on CSIDH do not only rely on the vectorization problem for their security, but also on some *variants* of this problem. Indeed, these variants can provide protocols with crucial additional functionality and enhanced efficiency. We note that this approach is reminiscent of a prior similar trend with DLP-based protocols (35 "DLP variants" are listed in [9]). However, as was the case with some DLP variants, one may wonder whether all vectorization problem variants used in CSIDH-based cryptographic protocols are equally hard [43].

In this paper, we focus on the *Vectorization Problem with Shifted Inputs* which underlies the security of CSI-SharK and BCP protocols [7, 8]. In DLP formalism, this variant provides an adversary not only with two group elements

g and g^x, but also with several other pairs (c, g^{x^c}). An algorithm to solve this problem for c in an interval $[0, M]$ would also solve the l-Diffie-Hellman inversion problem and the l-Strong Diffie-Hellman problem, which underlie the security of various DLP-based protocols [15–18, 30]. If the interval is punctured once, then it solves the n-Diffie-Hellman Exponent problem, used in protocols such as [19,35,44]. The group action version of this problem is also considered in [32], where it is proven secure in the general group action model.

So far, the only cryptanalytic work leveraging the extra information provided in the Vectorization Problem with Shifted Inputs is the classical attack of Kim [42], which adapts Cheon's algorithm [25] to the isogeny group action setting. This attack affects instances where a shift g^{x^c} is published such that c divides the order of the class group, and such instances are specifically avoided in the CSI-SharK and BCP key generation algorithms. Apart from this restriction, the security analysis and parameter selection in [7,8] are solely based on CSIDH. In particular, they ignore additional information provided in the Vectorization Problem with Shifted Inputs, and the protocols' quantum security is assumed to be equivalent to CSIDH quantum security.

Contributions. In this work, we apply a quantum *multiple* hidden shift algorithm of Childs and van Dam (CvD) [27] to the Vectorization Problem with Shifted Inputs. We carefully select subroutines to be used in the algorithm, and discuss different trade-offs between them. Among these subroutines is a knapsack problem, which we reduce to a Closest Vector Problem (CVP) instance in the ℓ_∞ norm. The literature on CVP algorithms for the ℓ_∞ norm is much sparser than for the Euclidean norm. We consider two approaches to solve this problem, respectively based on enumeration and sieving. We propose concrete algorithms taking inspiration from existing ones, but with crucial optimizations taking into account the application context. We also provide concrete (not asymptotic) complexity estimates for all of these subroutines, filling a gap in the literature.

Our results can be applied to any group action but we particularly focus on the isogeny-based group action for exposition since it is currently the most widely studied example of a post-quantum group action in cryptography.

As an example, we apply our analysis to the CSI-SharK and BCP protocols [7, 8]. For CSIDH-512 parameters and the suggested 2^{12} public keys, we estimate that the CSI-SharK and BCP protocols can be broken using between 2^{45} to 2^{57} T-gates, depending on whether the [11] or [21] cost model is used to model the group action evaluation[5]. This significantly improves over Peikert's 2^{54} to 2^{71} state-of-the-art estimates for the single public key instance. More generally, we quantify how the quantum security of *Vectorization Problem with Shifted Inputs* degrades when the number of public keys increases, via concrete estimates for both CSIDH-512 and CSIDH-1024 parameters (see Tables 3 and 4). We note that the best T-gate complexities are obtained with the sieving approach. On

[5] Note that [11] only targets T-gate counts whereas [21] (in its latest eprint version) also attempts to limit the number of qubits.

the other hand, the enumeration approach requires lower memory, which may result in a more practical algorithm for some quantum computer architectures.

While the complexity improvements we obtain are interesting on their own, our concrete analysis of Childs-van Dam's algorithm could also find further applications in cryptography, including of course in isogeny-based cryptography. We conclude the paper with potential improvements and extensions of this analysis. Various hard problems similar to Vectorization Problem with Shifted Inputs have appeared in the literature; as evidenced by our work (and previous work on DLP variants), their actual hardness may not follow from the hardness of CSIDH. We encourage the community to study their concrete quantum security beyond a mere application of Kuperberg's algorithms and variants.

Outline. In Section 2, we recall details of CSIDH and its state-of-the-art quantum cryptanalysis, as well as the CSI-SharK signature scheme and its hardness assumption. We describe the quantum algorithm by Childs and van Dam in Section 3. In Section 4, we show how a knapsack problem appearing in the Childs-van Dam algorithm can be formulated as a closest vector problem in the ℓ_∞ norm. We solve this problem using enumeration in Section 5 and sieving in Section 6. We give concrete complexity estimates for running the attack on CSI-SharK parameters and investigate how the attack improves for a larger number of shifts in Section 7. We conclude the paper and discuss several avenues of future work in Section 8.

2 CSI-SharK protocol

In this section, we recall the CSIDH group action, known quantum cryptanalysis approaches against it and the CSI-SharK protocol.

2.1 CSIDH group action

We briefly outline some key concepts about CSIDH and its underlying group action. For a more in-depth exposition of these topics see either CSIDH [24] or the textbook from Cox [28, Sect. 7].

Consider the set of supersingular elliptic curves defined over $\mathbb{F}_p$ whose $\mathbb{F}_p$-rational endomorphism ring is a particular order $\mathcal{O}$ in the quadratic imaginary field $K = \mathbb{Q}(\sqrt{\Delta})$, where Δ is the discriminant of the Frobenius characteristic polynomial. We will denote this set of elliptic curves by $\mathcal{E}\ell\ell_p(\mathcal{O})$. Then the *ideal class group* of $\mathcal{O}$ is the quotient of invertible fractional ideals, $I(\mathcal{O})$, and principal fractional ideals, $P(\mathcal{O})$, which we denote by $\mathrm{cl}(\mathcal{O}) = I(\mathcal{O})/P(\mathcal{O})$. Let $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$ be an ideal class. Then $\mathfrak{a}$ acts on any elliptic curve $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ via an isogeny

$$\varphi_\mathfrak{a} : E \to E/\mathfrak{a},$$

where $\ker(\varphi_\mathfrak{a}) = \cap_{\alpha \in \mathfrak{a}} \ker(\alpha)$. This gives rise to the following free and transitive group action

$$\mathrm{cl}(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}) \to \mathcal{E}\ell\ell_p(\mathcal{O}),$$

$$[\mathfrak{a}] \star E \mapsto E/\mathfrak{a}.$$

For brevity, we drop the $\star$ going forward and simply denote the group action as $[\mathfrak{a}]E$.

Equipped with this isogeny group action, the CSIDH protocol naturally follows the structure of a Diffie-Hellman key exchange [24].

A key algorithmic task to enable this scheme is of course the computation of the group action. Note that any ideal class $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$ has a representative that can be written as the product of smaller prime ideals. Without this decomposition into smaller ideals, the group action computation was otherwise infeasible to compute (at the time CSIDH was invented[6]). Furthermore, random sampling (which is necessary for safe key generation) requires either costly rejection sampling or an expensive computation in order to determine the structure of the class group. For the CSIDH-512 parameter set, this class group computation was done by the authors of CSI-FiSh [12].

As will be discussed later on, the exact cost of running a group action evaluation, especially on a quantum computer, may differ due to different space/time trade-offs.

2.2 Quantum cryptanalysis of CSIDH

The hidden shift problem. The works of Childs, Jao, and Soukharev [26] and Biasse, Jao, and Sankar [13] showed how to frame the hard problem underlying CSIDH as a hidden shift problem. Recall that given two curves $E, E' \in \mathcal{E}\ell\ell_p(\mathcal{O})$ we would like to find an ideal class group $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$ such that $E' = [\mathfrak{a}]E$. Then we can define two functions f_0, f_1 as $f_0 : [\mathfrak{b}] \mapsto [\mathfrak{b}]E$ and $f_1 : [\mathfrak{b}] \mapsto [\mathfrak{b}]([\mathfrak{a}]E)$. Notice that f_0 is injective and $f_1(\mathfrak{b}) = f_0(\mathfrak{b}\mathfrak{a})$, i.e. the functions f_0 and f_1 are equal up to a shift $\mathfrak{a}$. This means we can apply quantum hidden shift algorithms, like that of Kuperberg [45], to recover the secret $\mathfrak{a}$ in subexponential time.

Kuperberg's algorithm uses a complexity of $2^{O(\sqrt{\log N})}$ and works in any finite abelian group. An algorithm from Regev [54] gives a space improvement at the cost of a slower run time, but restricts to the $\mathbb{Z}_{2^n}$ case. Later, Kuperberg released a second algorithm in [46] that also benefited from a space improvement but this time in the general setting, called the *collimation sieve*. Peikert [53] built an attack on the CSIDH-512 parameter set using such a collimation sieve. He estimates that his attack requires between 2^{38} and 2^{47} T-gates, and an additional 2^{14}–2^{19} calls to the group action oracle. Note that to run any of these algorithms, it is necessary to have access to such a quantum oracle that can compute the group action. The exact cost of such an oracle is non-trivial to compute.

Quantum costs of computing the group action. Both the work of Bernstein, Lange, Martindale, and Panny [11], and that of Bonnetain and Schrottenloher [21] give estimates on the cost of evaluating a class group action for the CSIDH-512 parameter set on a quantum computer. The former work optimizes

[6] Clapoti(s), the recent work of Page and Robert [51], may mean that such costly computations can be avoided, though its direct impact has not been shown yet.

for the number of T-gates, while the latter considers how to balance with the number of qubits being used. Since there is some debate about the cost of quantum resources, this in part explains why there are different estimates for a (quantum) attack on CSIDH, making it difficult to estimate the security of its parameter sets.

The work of [11] focuses on giving in-depth analyses of the isogeny-related computations such as finding a point of fixed degree on an elliptic curve and computing ℓ-isogenous curves, using classical circuits. They choose to minimize the number of non-linear bit operations (thus minimizing the Toffoli and T-gates) at the cost of more qubits. Following the Bennett conversion [10], we can estimate that their algorithm for computing one iteration of the CSIDH-512 group action [11, Alg. 7.1] would require up to $2^{32.8}$ Toffoli gates, $2^{35.6}$ T-gates, and $2^{32.8}$ ancilla qubits.

The analysis from [21, Table 3] focuses on reducing the number of qubits necessary in their algorithm. They concluded that one group action call would require $2^{49.6}$ Toffoli gates, $2^{52.4}$ T-gates, and less than $2^{15.3}$ ancilla qubits.

2.3 CSI-SharK

CSI-SharK [7] is a threshold signature scheme that was adapted from CSI-FiSh [12]. Here, we first briefly recall the CSI-FiSh sigma protocol, and then explain how it was used to construct CSI-SharK.

CSI-FiSh sigma protocol. In this work Beullens, Kleinjung, and Vercauteren [12] build upon the ideas from Stolbunov to obtain a signature scheme using the CSIDH isogeny group action. The scheme starts from a standard proof of knowledge where you begin by fixing the starting curve to $E_0 : y^2 = x^3 + x$. Then the prover samples their secret, $\mathfrak{a} \in \mathrm{cl}(\mathcal{O})$, and computes their public key $E_A = [\mathfrak{a}]E_0$. To prove knowledge of their secret key to a verifier, they first send a commitment $E_1 = [\mathfrak{b}]E_0$. The verifier replies with a challenge $c \in \{0, 1\}$. If $c = 0$ then the prover sends $r = [\mathfrak{b}]$, and the verifier checks that $E_1 = [r]E_0$. Otherwise when $c = 1$, the prover sends $r = [\mathfrak{b}\mathfrak{a}^{-1}]$, and the verifier checks that $E_1 = [r]E_A$. The isogeny diagram for this scheme is outlined in Fig. 1.

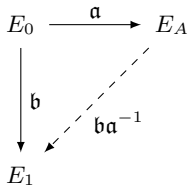

Fig. 1. The isogeny diagram related to the CSI-FiSh sigma protocol.

CSI-SharK. CSI-SharK expands the public key to allow for use in threshold signatures, and was based on the protocol from Baghery, Cozzo, and Pedersen (BCP) [8]. Their idea was to choose one secret exponent, but use it to generate several different public keys. First, fix a generator of the class group (or of a sufficiently large subgroup thereof), say $\mathfrak{g}$. Let N denote the size of the group generated by $\mathfrak{g}$. Then CSI-SharK uses public keys of the form

$$\left(E_0, E_1 = [\mathfrak{g}^{c_1 \cdot z}]E_0, \ldots, E_M = [\mathfrak{g}^{c_M \cdot z}]E_0,\ \{c_i\}_{i=1}^{M}\right).$$

Note that the $\{c_i\}_{i=1}^{M} \subset \mathbb{Z}_N$ are known, and $z \in \mathbb{Z}_N$ is the secret key. The set $\{c_i\}_{i=1}^{M}$ must have the special property of being a (super) *exceptional set*. That is, their pairwise differences (and sums) must be invertible modulo N. This property is essential for proving soundness. In particular, since c_0 is always chosen to be 0, this means that no c_i divides N. This avoids an attacker using something like Cheon's algorithm [25]. In this algorithm, Cheon uses an improved baby-step giant-step approach to recover the secret.

Therefore, the choice of $\{c_i\}_{i=0}^{M}$ is extremely important. In BCP, the c_i are chosen to be necessarily smaller than the smallest prime factor of N. If M is larger than the smallest factor, then they restrict to a subgroup $\mathbb{Z}_{N'}$, where $N'|N$ whose smallest prime factor is larger than M. The most natural choice, as proposed by the authors in BCP [8, Footnote 1], is the consecutive list of integers $\{0, \ldots, M\}$. In Table 1 of the same work, they list parameter sets for $M \in \{2^1, 2^2, 2^5, 2^8, 2^{10}, 2^{12}, 2^{15}, 2^{18}\}$. CSI-SharK [7, Table 2] suggests taking $M \in \{2^4, 2^8, 2^{12}\}$.

Both CSI-SharK and BCP rely on the hardness of the following problem.

Problem 2.1 ($(c_0, \ldots c_M)$-Vectorization Problem with Shifted Inputs). Given an element $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$, and the pairs $(c_i, [\mathfrak{g}^{c_i z}]E)_{i=0}^{M}$, where $\mathfrak{g}$ is fixed and $\mathbb{C}_M = \{c_0 = 0, c_1 = 1, c_2, \ldots, c_M\}$ is an exceptional set, find $z \in \mathbb{Z}_N$.

Whenever the starting curve E_0 has j-invariant 1728, we can compute the curve $E_i^t = [\mathfrak{g}^{-c_i z}]E_0$ from $E_i = [\mathfrak{g}^{c_i z}]E_0$ using the quadratic twist. Though this starting curve is not explicitly stated in [7], several of their optimisations take advantage of this easy twisting operation. As such, we may assume this starting curve implicitly. Note this gives us information about $2M - 1$ curves, instead of the M curves in the public key. For example, one of the proposed parameter sets in CSI-SharK set $M = 2^{12}$, and the $\{c_i\}_{i=0}^{M}$ are consecutive integers starting at 0. Including the twists, it means we have access to the following $2^{13} - 1$ curves

$$\{[\mathfrak{g}^{cz}]E_0 : c \in [-2^{12}, 2^{12}]\}.$$

Problem 2.1 was also considered in [32], where the authors standardize some notions and problems related to group actions and provide reductions between some of them. Currently, the state-of-the-art security analysis on Problem 2.1 is to use Kuperberg's algorithm on one single instance. The best known classical attack, described in CSIDH, is a meet-in-the-middle key search, also run on one single instance. These approaches fail to make use of any of the additional information provided.

3 Childs-van Dam algorithm

In this section we recall the Childs-van Dam algorithm [27], which will be the main quantum tool of our attack.

3.1 Algorithm

The algorithm from Childs and van Dam [27] aims at solving an instance of the *generalized hidden shift* problem.

Definition 3.1 (Generalized Hidden Shift). *For integers M, N, and finite set S, let $f : [-M, \dots, M] \times \mathbb{Z}_N \to S$ be a function that satisfies the following two criteria:*

(a) for fixed b, $f(b, x) : \mathbb{Z}_N \to S$ is injective;
(b) $f(b, x) = f(b + 1, x + z)$ for some fixed $z \in \mathbb{Z}_N$ and $b = 1, \dots, M - 2$.

The generalized hidden shift problem *asks to recover z given oracle access to f.*

In particular, (a) and (b) of Definition 3.1 imply that $f(b, x) = f(0, x - bz)$.

Note that we took $[-M, \dots, M] \times \mathbb{Z}_N$ as the domain of f, whereas related literature defines the problem often on the domain $[0, \dots, M'] \times \mathbb{Z}_N$ for some M'. However, these are clearly equivalent problems after relabeling the interval.

Note that when $M = 2$, Definition 3.1 is the standard *hidden shift* problem, and when $M = N$ it is the *hidden subgroup* problem which tries to detect $\ker(f) = \langle (1, z) \rangle$ in $\mathbb{Z}_N^2$.

Retrieving z from a CSI-Shark instance can be seen as solving a generalized hidden shift problem for the function f sending $(b, x) \in \{-M, \dots, M\} \times \mathbb{Z}_N$ to $f(b, x) = [\mathfrak{g}^x]E_{-b}$. Indeed we have

$$f(b, x) = [\mathfrak{g}^x]E_{-b} = [\mathfrak{g}^x][\mathfrak{g}^{(-b)z}]E_0 = [\mathfrak{g}^{x-bz}]E_0 = f(0, x - bz)$$

hence $f(b, x) = f(b + 1, x + z)$ as required.

We outline the approach by Childs and van Dam in Algorithm 1, and provide further explanation in what follows.

Notation. For the rest of this section, we call O_f an operator such that $O_f(|b\rangle |x\rangle |0\rangle) = |b\rangle |x\rangle |f(b, x)\rangle$. We note $k := \left\lceil \frac{\log(N)}{\log(2M+1)} \right\rceil$. For any $\alpha \in \mathbb{Z}_N$ and any $y \in \mathbb{Z}_N^k$, we define the set

$$S_\alpha^{\boldsymbol{y}} := \{\boldsymbol{i} \in \{-M, \dots M\}^k : \langle \boldsymbol{i}, \boldsymbol{y} \rangle = \alpha \pmod N\},$$

where $\langle \cdot, \cdot \rangle$ is the standard dot product. That is, S_α^y contains all solutions $\boldsymbol{i} \in \{-M, \dots, M\}^k$ to the knapsack problem $\langle \boldsymbol{i} \cdot \boldsymbol{y} \rangle = \alpha \pmod N$. We write O_C for an operator such that $O_C(|0, ..., 0\rangle |\alpha\rangle |y\rangle) = |C_y(\alpha)_1, \dots, C_y(\alpha)_k\rangle |\alpha\rangle |y\rangle$, where $(C_y(\alpha)_1, \dots, C_y(\alpha)_k)$ is in S_α^y with (high) probability P_{succ}^C. We will build the operator O_C as a reversible circuit C that returns a single tuple $(C_y(\alpha)_1, \dots, C_y(\alpha)_k)$ from α and $\boldsymbol{y}$.

Algorithm 1 Quantum algorithm of Childs and van Dam [27]

Input: N, M and *superposition* oracle access to f
Output: the hidden period z

1: Start with $k = \left\lceil \frac{\log(N)}{\log(2M+1)} \right\rceil$ groups of three registers of the form $|0,0,0\rangle$ where the three registers will respectively contain elements in $\{-M,\dots,M\}$, $\mathbb{Z}_N$ and S.
2: **for each** group of registers :
3: Apply the Quantum Fourrier Transform (QFT) over $\mathbb{Z}_M$ on the first register
4: Apply the QFT over $\mathbb{Z}_N$ on the second register
▷ The state is $\frac{1}{\sqrt{N(2M+1)}} \sum_{i\in\{-M,\dots,M\}} \sum_{x\in\mathbb{Z}_N} |i,x,0\rangle$.
5: Apply the oracle O_f on the registers
▷ The state is $\frac{1}{\sqrt{N(2M+1)}} \sum_{i\in\{-M,\dots,M\}} \sum_{x\in\mathbb{Z}_N} |i,x,f(i,x)\rangle$.
6: Measure the last register and get $c = f(0,x_0)$ for an unknown x_0
▷ The state is $\frac{1}{\sqrt{2M+1}} \sum_{i\in\{-M,\dots,M\}} |i, x_0 + iz\rangle$
7: Apply the QFT over $\mathbb{Z}_N$ on the second register
▷ The state is $\frac{1}{\sqrt{N(2M+1)}} \sum_{i\in\{-M,\dots,M\}} \sum_{y\in\mathbb{Z}_N} \omega^{y(x_0+iz)} |i,y\rangle$ where $\omega = e^{\frac{-2i\pi}{N}}$.
8: Measure the last register and get y (uniformly)
▷ The state is $\frac{1}{\sqrt{M}} \sum_{i\in\{-M,\dots,M\}} \omega^{yiz} |i\rangle$.
9: **end for**
▷ The global state is $\bigotimes_{j\in\{1,\dots,k\}} \left(\frac{1}{\sqrt{2M+1}} \sum_{i\in\{-M,\dots,M\}} \omega^{y_j iz} |i\rangle \right) = \frac{1}{\sqrt{2M+1}^k} \sum_{i_1,\dots,i_k\in\{-M,\dots,M\}} \omega^{z(\sum_{j=1}^k i_j y_j)} |i_1,\dots,i_j\rangle$.
10: Compute $\alpha = \sum_{j=1}^k i_j y_j \bmod N$ in a new register
▷ The global state is $\frac{1}{\sqrt{2M+1}^k} \sum_{i_1,\dots,i_k\in\{-M,\dots,M\}} \omega^{z\alpha} |i_1,\dots,i_j,\alpha\rangle$.
11: Apply O_C^{-1}, the reverse of the operator O_C
▷ The global state is "close"(detailed in the analysis) to $\frac{1}{\sqrt{N}^k} \sum_{\alpha\in\mathbb{Z}_N} \omega^{z\alpha} |0,\dots,0\rangle |\alpha\rangle$.
12: Apply the inverse of the QFT over $\mathbb{Z}_N$ to the last register
▷ The global state is "close" to $|0,\dots,0\rangle |z\rangle$.
13: Return the state and verify that it is $|0,\dots,0\rangle |z\rangle$ by checking the equality $f(0,0) = f(1,z)$ and retry if it is not.

Analysis of Algorithm 1. From Steps 2 to 9, we build k groups of registers with similar patterns to Kuperberg's algorithm. We start by building (through Steps 3 to 5) the uniform superposition of inputs and the associated output

$$\frac{1}{\sqrt{N(2M+1)}} \sum_{i\in\{-M,\dots,M\}} \sum_{x\in\mathbb{Z}_N} |i,x,f(i,x)\rangle .$$

We then measure the output $c = f(0,x_0)$ and by the periodic property of f, we get

$$\frac{1}{\sqrt{2M+1}} \sum_{i\in\{-M,\dots,M\}} |i, x_0 + iz\rangle .$$

We apply a QFT on the second register and measure it to have

$$\frac{1}{\sqrt{2M+1}} \sum_{i\in\{-M,\ldots,M\}} \omega^{yiz} |i\rangle \text{, where } \omega = e^{\frac{-2i\pi}{N}}.$$

Considering the k different registers together, the state can be written as

$$\bigotimes_{j\in\{1,\ldots,k\}} \left(\frac{1}{\sqrt{2M+1}} \sum_{i\in\{-M,\ldots,M\}} \omega^{y_j iz} |i\rangle \right) = \frac{1}{(\sqrt{2M+1})^k} \sum_{i_1,\ldots,i_k\in\{-M,\ldots,M\}} \omega^{z\left(\sum_{j=1}^k i_j y_j\right)} |i_1,\ldots,i_j\rangle$$

We compute $\alpha = \sum_{j=1}^k i_j y_j \bmod N$ in a new register and the global state becomes

$$\frac{1}{\sqrt{2M+1}^k} \sum_{i_1,\ldots,i_k\in\{-M,\ldots,M\}} \omega^{z\alpha} |i_1,\ldots,i_j,\alpha\rangle .$$

If we could erase $i_1,\ldots,i_k$ in the first k registers, we would get the state $\frac{1}{\sqrt{N^k}} \sum_{\alpha\in\mathbb{Z}_N} \omega^{z\alpha} |\alpha\rangle = QFT(|z\rangle)$, and recover the hidden shift z exactly. To approximate this result, we "uncompute" the registers $i_1,\ldots,i_k$, i.e., we reverse the operations of an operator that does

$$|0,\ldots,0\rangle |\alpha\rangle \mapsto \left(\frac{1}{\sqrt{|S_\alpha^{\boldsymbol{y}}|}} \sum_{(i_1,\ldots,i_k)\in S_\alpha^y} |i_1,\ldots,i_k\rangle \right) |\alpha\rangle ,$$

where we recall that

$$S_\alpha^{\boldsymbol{y}} = \{i_1,\ldots,i_k \in \{-M,\ldots,M\} | \sum_{j=1}^k i_j y_j = \alpha \bmod N\}. \tag{1}$$

This is approximated by applying O_C^{-1}, the inverse operator of O_C.

The probability of success of the algorithm (the probability of getting $|0,\ldots,0\rangle |z\rangle$ at the end of the procedure) is given by the dot product of the output of the procedure and $|0,\ldots,0\rangle |z\rangle$, namely:

$$\begin{aligned}
&\Pr(\text{success}) \\
&= \left| \left\langle (Id \otimes QFT^{-1}) \circ O_C^{-1} \left(\frac{1}{\sqrt{M}^k} \sum_{i_1,\ldots,i_k\in\{-M,\ldots,M\}} \omega^{z\alpha} |i_1,\ldots,i_j,\alpha\rangle \right) \middle| |0,\ldots,0\rangle |z\rangle \right\rangle \right|^2 \\
&= \left| \left\langle \frac{1}{\sqrt{2M+1}^k} \sum_{i_1,\ldots,i_k\in\{-M,\ldots,M\}} \omega^{z\alpha} |i_1,\ldots,i_j,\alpha\rangle \middle| O_C \circ (Id \otimes QFT) (|0,\ldots,0\rangle |z\rangle) \right\rangle \right|^2 \\
&= \left| \left\langle \frac{1}{\sqrt{2M+1}^k} \sum_{i_1,\ldots,i_k\in\{-M,\ldots,M\}} \omega^{z\alpha} |i_1,\ldots,i_j,\alpha\rangle \middle| \frac{1}{\sqrt{N}} \sum_{\alpha\in\mathbb{Z}_N} |C_y(\alpha)_1,\ldots,C_y(\alpha)_k\rangle |\alpha\rangle \right\rangle \right|^2
\end{aligned}$$

$$= \left| \frac{1}{\sqrt{N(2M+1)^k}} \sum_{\alpha \in \mathbb{Z}_N} \underbrace{\left\langle \sum_{i_1,\ldots,i_k \in S_\alpha} |i_1,\ldots,i_k\rangle \middle| |C_y(\alpha)_1,\ldots,C_y(\alpha)_k\rangle \right\rangle}_{1 \text{ if } (C_y(\alpha)_1,\ldots,C_y(\alpha)_k) \in S_\alpha^y} \right|^2$$

$$= \left| \frac{N}{\sqrt{N(2M+1)^k}} \Pr_{\alpha \in \mathbb{Z}_N} [C_y(\alpha)_j \in S_\alpha^y] \right|^2$$

$$\Pr(\text{success}) = \frac{N}{(2M+1)^k} (P_{succ}^C)^2 \tag{2}$$

3.2 Step 11: a knapsack problem

It remains to describe how to perform Step 11 of the algorithm, namely how to implement the operator O_C. Clearly, this amounts to solving (for a superposition of α values) the knapsack problem given by Equation (1).

Childs and van Dam use an integer programming algorithm due to Lenstra [37] to solve this problem. The algorithm seeks to determine whether there exists a solution, but can be adapted to return all possible solutions. This algorithm, as applied to the hidden shift application, runs in time $2^{O(k^3)}$. The authors mention that a result from Kannan [39] can be used for an improvement on this subroutine that runs in time $2^{O(k \log k)}$.

These asymptotic estimations for solving the integer programming problem at hand (implementing O_C) are useful for studying the asymptotic behavior of the Childs-van Dam algorithm, but they provide little insight on the actual cost of an attack against a concrete system such as CSI-SharK. For this reason we will reframe the knapsack problem as lattice problems in Sections 5 and 6, and evaluate explicit costs for running the overall attack.

3.3 Complexity analysis

Our complexity analysis of the Childs-van Dam (CvD) algorithm will depend on three subroutines : the quantum subroutine that performs the (isogeny) group action in superposition, Quantum Fourier Transforms, and solving the knapsack problem from Step 11.

Group action cost. Recall from Section 2.2 that there is some debate about the actual quantum cost of evaluating the isogeny group action, depending on which cost metric is being optimized: T-gate count or qubit count. Thus we recall the complexities here, but we will keep this portion of the complexity analysis modular so that it is easy to modify according to which estimate is being considered. In [11], Bernstein, Lange, Martindale, and Panny give an analysis on the quantum security of CSIDH. In doing so, they improve upon isogeny computation

algorithms with the goal of minimizing the T-gate cost of the group action. The estimate from [11, Alg. 7.1] for the CSIDH-512 group action is 2^{40} non-linear bit operations, giving at least 2^{40} quantum operations, and 2^{29} ancilla qubits. Similarly, Bonnetain and Schrottenloher also give a security analysis of CSIDH in [21] but place more emphasis on balancing the amount of quantum memory and the number of qubits necessary, together with the T-gate count. They present several trade-offs between three quantum abelian hidden shifts algorithms, and focus on tweaking these quantum algorithms to gain improvements (as opposed to tweaking the isogeny algorithms as in [11]). In [21, Table 3] they concluded that one group action call for CSIDH-512 would require $2^{52.4}$ quantum operations and less than $2^{15.3}$ ancilla qubits.

Quantum Fourier Transforms. In [3], Ahokas, Cleve, and Hales give a new upper bound on the number of quantum gates necessary to compute a Quantum Fourier Transform modulo 2^n. They improved upon the previous upper bound of $O(n \log n)$, to $O(n(\log \log n)^2 \log \log \log n)$. This cost, however, is negligible in Childs-van Dam's algorithm, so we use the "naive" estimate of $n(n+1)/2$ gates.

QRAM costs. Quantum random-access memory (QRAM) is a circuit that allows a quantum algorithm to access some classical data stored in memory. QRAM is necessary in many important quantum algorithms and will be pertinent to our analysis later on. Since QRAM sometimes constitutes the bottleneck of some algorithms, there have been different proposals for how to model such a circuit. In [49], Di Matteo, Gheorghiu, and Mosca model (among others) a type of circuit known as the *bucket brigade circuit*. They give a rough cost estimate for storing classical memory with n-bit addresses by multiplying (# logical qubits) $\times$ (T-depth). In total the bucket brigade circuit run in parallel had a cost of $O(n \cdot 2^n)$.

Quantum complexity analysis of Childs-van Dam's algorithm. Recall from Algorithm 1 that $k := \left\lceil \frac{\log(N)}{\log(2M+1)} \right\rceil$. The algorithm uses 3 QFTs in Steps 3, 4, 7 per loop. This loop also includes one call to the group action oracle in Step 5. Step 10 consists of some computations which we claim are negligible compared to the rest of the computations. Finally, Step 11 describes a knapsack problem and in Step 12 we have one final (inverse) QFT to do. Thus, the algorithm requires a total quantum operation count of

$$(3k+1) \text{ QFTs } + k \text{ group action evaluations } + \text{ knapsack problem.}$$

Note that k is relatively small in our application (for CSI-SharK parameters, we will have $k = 20$). Thus, this algorithm differs from prior cryptanalysis in that the number of group action evaluations is small. For example, the algorithm from Peikert's quantum analysis of CSIDH [53, Fig. 2] for the same parameter sizes requires between 2^{14} to 2^{20} group action evaluations, depending on the length of the phase vector. Recall that the group action evaluation is very expensive, so this reduction in the number of evaluations can have a significant impact on the overall cost.

In what follows, we evaluate the cost of solving the knapsack problem, before concluding the paper with concrete complexity estimates for the attack.

4 Solving the knapsack problem

Here we show how the knapsack problem in the Childs-van Dam algorithm can be formulated as an instance close to a Closest Vector Problem (CVP) in a particular lattice. The key difference will be that we will require *all* of the solutions up to a certain bound. Additionally, this instance of CVP will use the ℓ_∞ norm instead of the usual ℓ_2 (Euclidean) norm. We describe how to actually solve the resulting CVP instance in the subsequent sections.

4.1 CVP formulation

Recall that to complete our attack we must compute the set

$$S_\alpha^{\boldsymbol{y}} = \{\boldsymbol{i} \in C^k : \langle \boldsymbol{i}, \boldsymbol{y} \rangle = \alpha \mod N\},$$

where $C = \{-M, \ldots, M\}$ is the set of available shifts we have access to, and k is the dimension we choose such that $|C^k| = (2M+1)^k \approx N$. Note that $\boldsymbol{y}$ is fixed at the beginning of Step 11, but can be re-randomized in our application by repeating Steps 1-10. Further, α takes all possible values in superposition, and $\boldsymbol{i}$ are the variables we are solving for.

In [27], the set $S_\alpha^{\boldsymbol{y}}$ is computed using a classical integer programming algorithm due to Lenstra [37]. This computation can be efficient when C is taken to be sufficiently large, which is not always the case. Furthermore, the integer programming computation that was originally proposed is aimed at solving the worst case instance and does not offer concrete complexity estimates, making it difficult to measure the overall security of target systems. For these reasons we proceed with a reduction to an equivalent lattice problem.

In $S_\alpha^{\boldsymbol{y}}$, we are considering solutions $\boldsymbol{i} \in C^k$ to the equation

$$\langle \boldsymbol{i}, \boldsymbol{y} \rangle = \alpha \mod N. \tag{3}$$

We can construct a lattice, L_y, from $\boldsymbol{y}$ where the rows in the following matrix represent the basis vectors of L_y. Without loss of generality, suppose that y_1 is invertible; otherwise, we can use any other y_j that was invertible or rerun Steps 1 - 10 in Algorithm 1 until some y_j is invertible. We can then divide the rows through by y_1, giving a simplified version of the matrix in the form

$$\begin{bmatrix} y_2' & 1 & & 0 \\ \vdots & & \ddots & \\ y_k' & & & 1 \\ N & 0 & \cdots & 0 \end{bmatrix},$$

where $y_i' = y_i/y_1 \mod N$. Thus, vectors in $L_y$will be such that the first component is a linear combination of the y_i up to a multiple of N, and the rest of the components encode information about the linear combination.

Now consider

$$\begin{bmatrix} i_2 \cdots i_k \ \lambda \end{bmatrix} \begin{bmatrix} y_2' & 1 & & 0 \\ \vdots & & \ddots & \\ y_k' & & & 1 \\ N & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} \alpha' \ 0 \cdots 0 \end{bmatrix} + \begin{bmatrix} -i_1 \ i_2 \cdots i_k \end{bmatrix}, \tag{4}$$

where $y_i' = y_i/y_1 \mod N$ and $\alpha' = \alpha/y_1 \mod N$. We see that the first component in this system will be exactly Eq. (3), and the rest of the components are trivially of the form $i_t = i_t, t \in [2, k]$. This shows that the CVP target vector $\begin{bmatrix} \alpha' \ 0 \cdots 0 \end{bmatrix}$ is close to the lattice L_y, in the sense that it is a lattice vector plus some small noise $\begin{bmatrix} i_1 \ -i_2 \cdots -i_k \end{bmatrix}$. Specifically, since S_α^y requires $\boldsymbol{i} \in C^k$, this imposes the restriction that each component of $\boldsymbol{i}$ be less than M. Recall that the ℓ_∞ norm of a vector $\boldsymbol{x} = [x_1, \ldots x_n]$ is defined as $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$. So the small noise is bounded with $\|\boldsymbol{i}\|_\infty < M$.

4.2 CVP specificities

Note that the elements $\{y_i\}$ defining the lattice L_y are computed in Step 7 of Algorithm 1. This involves a QFT and so must be done quantumly. The remaining steps in the algorithm amount to solving a knapsack problem, which we have now formulated as an (almost) CVP instance, and the final QFT measurement. It is not *exactly* a CVP since we will be searching for all of the solutions within a fixed bound, not just one. For ease of explanation, we will abuse notation and keep the term CVP. Further, it must be performed in superposition since α is in superposition; however, some classical preprocessing after the elements $\{y_i\}$ are measured is also possible. In Sections 5 and 6 we will consider two options for solving this CVP instance.

As mentioned already, we observe that we can repeat the first 10 steps of Algorithm 1 until a favourable basis $\{y_1, \ldots y_k\}$ is found. Note that while α is in superposition, the choice of basis is fixed. This means that we can continue our analysis assuming that both the lattice reduction and the CVP instance are average cases instead of worst cases. We will keep this in mind when choosing which CVP solver to use, as this could lead to savings in the overall complexity. The approach from Section 5 will be an example where having a short basis (in the ℓ_∞ norm) will have a big impact on the final concrete complexity of the overall computation.

4.3 Selecting CVP solvers

Our main task going forward will be to solve an instance of CVP in the ℓ_∞ norm. Most of the literature describing algorithms for solving CVP problems targets the ℓ_2 norm, so we will need to tailor our approach to fit our case.

Furthermore, as discussed in Section 4.2, we will only be concerned with the average case complexity of the algorithm, not the worst case. We categorize the current literature into three main approaches that solve CVP problems: enumeration [38, 39], sieving [1, 4], and Voronoi sets [33, 50].

Kannan [39] gives an enumeration algorithm for solving integer programming problems that splits the problem into several smaller ones. The time complexity of the algorithm is $2^{O(n \log n)}$. It is also deterministic and there is no space complexity. In Section 5 we detail a novel variant of such an enumeration approach that is tailored to the setting of our problem. It requires little memory and is easy to implement, but the asymptotic time complexity will be worse than other approaches, meaning it may not scale so well to larger parameter sets.

Later on, in Section 6 we focus on a sieving approach. In [4], Ajtai, Kumar, and Sivakumar give a randomized algorithm based on sampling and sieving. The work of Aggarwal and Mukhopadhyay [1] follows the strategy from [4] at a high level, but focuses on the special case of using the ℓ_∞ norm. This approach offers a better time complexity than the enumeration approach, but will also require exponential size quantum memory. Recall from the discussion in Section 3.3 that QRAM is very costly so this will be a serious caveat of sieving.

We do not provide an approach using Voronoi sets, in part because the single exponential time complexity estimates computed in [50] are only for the Euclidean norm. In [14], the authors show that for non-Euclidean norms, it is unlikely to find algorithms of single exponential space and time that can solve CVP. They show that for some other norms the case is *similar* to the Euclidean one, and so may have single exponential algorithms. It is not clear how the ℓ_∞ norm plays into this, so instead we leave comparing such an approach (and any others) to future work.

5 Instantiating the attack with enumeration

As described in Section 4.1, the problem which remains to be solved to complete our attack using the algorithm by Childs and Van Dam can be seen as a CVP problem in the ℓ_∞ norm. More precisely, for any $\boldsymbol{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_N^k$, solutions to Eq. (3) correspond to those $\boldsymbol{i}$ with distance at most M to a target vector in ℓ_∞ norm satisfying Eq. (4). In this section, we describe a simple method which solves this CVP problem using enumeration, then we evaluate the cost of this approach in the context of our attack using the Childs-van Dam algorithm.

5.1 Solving the ℓ_∞ CVP with enumeration

Given a fixed lattice with basis given by the matrix A and a target vector b defined using α, we want to find all points in the lattice that are at a distance at most M from b in the ℓ_∞ norm.

Let $x_0 := \lfloor A^{-1}b \rceil$ be the coordinates of the target vector with respect to the basis of A rounded to the nearest integer, i.e. $\|A^{-1}b - x_0\|_\infty \leq \frac{1}{2}$. Assume now that x is a vector that is a solution to our CVP problem at hand, i.e. we have

$\|Ax - b\|_\infty \leq M$. While x_0 will in general not be a solution to the CVP problem itself, it approximates solutions since

$$\begin{aligned}\|x - x_0\|_\infty &\leq \|x - A^{-1}b\|_\infty + \|A^{-1}b - x_0\|_\infty \\ &\leq \|A^{-1}\|_\infty \cdot \|Ax - b\|_\infty + \|A^{-1}b - x_0\|_\infty \\ &\leq ||A^{-1}||_\infty \cdot M + \frac{1}{2}.\end{aligned}$$

This observation allows us to find *all* possible solutions in the lattice spanned by A by enumerating through all x that lie at distance at most $\|A^{-1}\|_\infty \cdot M + \frac{1}{2}$ from x_0. This gives at most $2\|A^{-1}\|_\infty \cdot M + 2$ options for each coordinate of x (from the positive and negative directions and 0), hence a total of $(2\|A^{-1}\|_\infty \cdot M + 2)^k$ choices. For each such x, we can verify whether it is a solution by checking whether $\|Ax - b\|_\infty$ is at most M. This check costs at most one matrix vector multiplication and one vector addition.

Algorithm 2 Algorithm using enumeration to solve CVP in ℓ_∞

Input: Basis A of full-rank k-dimensional lattice, target vector b.
Output: Lattice vectors with distance at most M from b with respect to ℓ_∞ norm.

```
Compute x_0 := ⌊A^{-1} b⌉.
for each x with ‖x − x_0‖_∞ ≤ ‖A^{-1}‖_∞ · M + 1/2
    if ‖Ax − b‖_∞ ≤ M
        Ax is a close vector.
    end if
end for
Return all close vectors found.
```

Lemma 5.1. *Algorithm 2 finds all vectors at distance at most M with respect to the ℓ_∞ norm in the k-dimensional lattice with basis A after enumerating at most $(2\|A^{-1}\|_\infty \cdot M + 2)^k$ vectors.*

Reducing enumeration costs by reducing $\|A^{-1}\|_\infty$. Given that the number of vectors we have to enumerate depends on $\|A^{-1}\|_\infty$, it makes sense to preprocess the basis A to lower $\|A^{-1}\|_\infty$, e.g. using LLL to obtain a basis with entries of roughly the same size, before starting the enumeration.

For a typical reduced basis we can heuristically expect all entries of A to be roughly of size $\det(A)^{1/k} = N^{1/k}$ up to a constant c_k which depends on the dimension of the lattice and will be small for the dimensions considered in this paper. Similarly, when reducing A^{-1}, we can expect its entries to be about $c_k \cdot \det(A^{-1}) = c_k \cdot N^{-1/k}$. This means the norm $\|A^{-1}\|_\infty$ will be about $c_k \cdot k \cdot N^{-1/k}$. For $M < N^{1/k}$, the cost estimate of Lemma 5.1 can thus be very roughly upper bounded by $(2 \cdot c_k \cdot k + 2)^k$.

Reversely, we can also get a lower bound for the complexity of Algorithm 2.

Lemma 5.2. *To find all lattice vectors of distance at most M with respect to the ℓ_∞ norm in the k-dimensional lattice defined by the matrix A, Algorithm 2 requires to iterate through at least 4^k vectors.*

Proof. Minimizing A^{-1} in the ℓ_∞ norm is equivalent to minimizing the dual basis of A with respect to the ℓ_1 norm. The ℓ_1 norm is at least as large as the ℓ_2 norm, which can be bounded from below by $\det(A)^{-1/k}$ using Hadamard's inequality. Now the claim follows from Lemma 5.1.

We will discuss further approaches to reduce the value $\|A^{-1}\|_\infty$ in the context of Childs-Van Dam's algorithm in Section 5.3.

Amplitude amplification. We briefly discuss how to use Grover's search to accelerate the enumeration process of Algorithm 2. Consider amplitude amplification introduced by Brassard, Hoyer, Mosca and Tapp [23], which extends Grover's algorithm [36] to any search space. Algorithm 3 shows how Algorithm 2 can be modified using this amplitude amplification.

Algorithm 3 Algorithm using enumeration to solve CVP in ℓ_∞

Input: Basis A of full-rank k-dimensional lattice, target vector b.
Output: Lattice vector with distance at most M from b with respect to infinity norm.
1: Compute $x_0 := \lfloor A^{-1}b \rceil$.
2: **Grover search** on x with $\|x\|_\infty \leq \|A^{-1}\|_\infty \cdot M + \frac{1}{2}$ with $\frac{\pi}{4}(2\|A^{-1}\|_\infty \cdot M + 2)^{k/2}$ turns using the following oracle:
3: Check if $\|A(x + x_0) - b\|_\infty \leq M$
4: **EndGrover**
5: Return all close vectors found.

The classical enumeration cost of $(2\|A^{-1}\|_\infty \cdot M + 2)^k$ vectors becomes $\frac{\pi}{2}(2\|A^{-1}\|_\infty \cdot M + 2)^{k/2}$ steps of amplification using [23, Thm. 4]. In each step of the amplification, we have to generate the superposition of x, which can be done by generating all of its k components individually using a QFT of size $(2\|A^{-1}\|_\infty \cdot M + 2)$ and checking if $\|A(x + x_0) - b\|_\infty \leq M$. The latter requires at most k^2 multiplications in $\mathbb{Z}_N$.

Remark 5.3. Classically the enumeration algorithm returns all close enough vectors. On the other hand, amplitude amplification will only allow to find a single solution faster. Note that this is sufficient to solve the problem at hand, and also that we set the dimension of the lattice used such that we only expect a single solution.

5.2 Applying the enumeration to Childs-Van Dam's algorithm to attack CSI-SharK

Next, we use the enumeration approach from Section 5.1 when applying the Childs-Van Dam algorithm to cryptanalyze CSIDH variants such as CSI-SharK.

Recall that applying Childs-Van Dam to the setting of CSI-SharK, the quantum computations up to Step 10 provide us with a CVP problem in the infinity norm. More precisely, the computations up to Step 10 in CvD provide us with a basis of a lattice, A, and a target vector, b. Hereby, A is provided after measurement and can therefore be processed classically or quantumly, whereas the target vector b, defined using α, is only available in superposition. For the parameters provided in CSI-SharK we are left to solve a CVP problem in ℓ_∞ norm for the given lattice and target vector, where solutions are at distance at most M as was described in Section 4.1. By Lemma 5.1, this can be achieved by iterating through at most $(2\|A^{-1}\|_\infty \cdot M + 2)^k$ vectors in the lattice. Given the dependence of the enumeration's complexity on $\|A^{-1}\|_\infty$, there are two ways of optimising this enumeration approach:

1. Preprocess A as previously mentioned (see Section 5.3) to lower $\|A^{-1}\|_\infty$.
2. Repeat the initial quantum computation from Step 1 to Step 10 until the measured lattice basis A is good, i.e. $\|A^{-1}\|_\infty$ is low.

Both steps together make our strategy. We repeat the initial quantum computation to get new random lattices, and we reduce their bases in order to minimize $\|A^{-1}\|_\infty$. If the resulting basis is sufficiently good, i.e. $\|A^{-1}\|_\infty$ of the resulting basis A is smaller than a desired threshold, we move forward and run the enumeration, otherwise we sample another lattice.

Note that each time we repeat the initial steps of the quantum computation, we obtain a new random lattice in the set of lattices of the form Eq. (4) with entries $y_i' \in \mathbb{Z}$. We summarise the resulting strategy in Algorithm 4.

Define $c_A := \|A^{-1}\|_\infty \cdot N^{1/k}$, where A corresponds to a basis of a k-dimensional lattice L_y. From Section 5.1, we know that $c_A \geq 1$ and that we would expect $c_A \leq k$ for a reduced basis. In Section 5.3, we will give experimental results on the size of c_A for lattices appearing in the setting of CSI-SharK parameters. Further, we present experiments that show how much we expect c_A to decrease when preprocessing the basis and sampling a new lattice if the resulting basis is not sufficiently "good". When fine tuning the attack for CSI-SharK, these experiments are crucial to determining what values for c_A we can expect.

Note that for an appropriately chosen threshold for c_A, the preprocessing of the lattice will provide a basis A satisfying this threshold and Algorithm 2 will always terminate (see Section 5.3 for reasonable values of c_A). For the version of the enumeration using Grover's algorithm, note that each execution of the while loop in Algorithm 4 runs the initial steps of Algorithm 1, until Step 10, which costs essentially k group action evaluations. Afterwards, the algorithm needs to enumerate at most $(2c_A \cdot \frac{M}{N^{1/k}} + 2)^k$ vectors by Lemma 5.1.

5.3 Improving the complexity of the enumeration by minimizing c_A

The preceding subsections raised the question of how to preprocess a basis of a lattice to minimise c_A in order to speed up the enumeration and the question of how c_A is distributed when sampling new random lattices of the shape Eq. (4).

Algorithm 4 Using Enumeration to find all close vectors for Childs-Van Dam algorithm.

Input: Class number N, M and superposition oracle access to f and threshold t for $\|A^{-1}\|_\infty$.

Output: All vectors of distance at most M from target vector required by Childs-Van Dam algorithm.

1: Let $c_A > t$.
2: **while** $c_A > t$
3: Run quantum algorithm up to Step 10 which gives lattice L_y classically and target vector b in superposition.
4: Preprocess basis of L_y to minimise $\|A^{-1}\|_\infty$ and compute resulting c_A.
5: **end while**
6: Run Algorithm 2 to get all vectors in L_y at ℓ_∞ distance at most M from α.
7: Return all close vectors.

Note that after measuring the lattice basis, any preprocessing of the lattice in Algorithm 4 can be done entirely using classical operations. While this may or may not be implemented on a quantum computer in the end, in any case we can implement and test this part of the algorithm on a classical computer. We implemented one method to preprocess the lattice and will present our experimental results in the following. More precisely, we computed an approximation of a good basis by reducing the basis of the dual lattice using LLL in the ℓ_2 norm – partially motivated by the fast implementations of LLL available. Then, we reduced $\|A^{-1}\|_\infty$ of the resulting lattice with respect to the infinity norm using an algorithm due to Lovasz and Scarf [47]. Note that this final use of the Lovasz-Scarf algorithm can only improve upon the result after the LLL reduction, and is expected to give an improvement since in general a small linear combination of a good ℓ_2 basis can also reduce the ℓ_1 norm.

Using the class number $N \approx 2^{256}$ and the dimension k sufficiently large to run the attack, i.e. $k = 20$, we generated random lattices of the form as in Eq. (4) by sampling $y'_i \in \mathbb{Z}_N$ uniformly at random, consistently with the distribution produced in Step 8 of Algorithm 1. When using the approach outlined above to preprocess the lattices using LLL and Lovasz-Scarf, we experimentally obtained $c_A \approx 6.3$ on average[7]. By discarding lattices that do not lead to small c_A, and sampling a new lattice, this constant can be improved. Fig. 2 shows an experiment of the minimum observed value for c_A plotted against the number of randomly chosen lattices, where the basis A is obtained after LLL- and Lovasz-Scarf-reducing the dual basis. After a few 1000 lattices, the minimal c_A observed reliably reaches ≈ 5.2. See Table 1 for further experimental results regarding c_A for different values of M.

As one might expect, the minimum ℓ_∞ norm observed decreases with the number of lattices tried. With an exhaustive search over (short) linear combi-

[7] The code to run the experiments, implemented in Magma [22], is provided at `https://anonymous.4open.science/r/CvD-analysis-A4E7/`. Experimental results for different parameters are shown in Table 1.

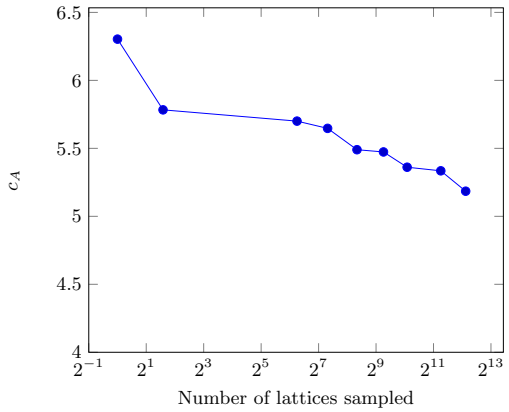

Fig. 2. Minimal c_A observed using LLL- and Lovasz-Scarf reduction vs the number of random lattices of the form Eq. (4) sampled for $k = 20$.

nations of lattice vectors we were then able to sometimes further improve this constant (at an exponential cost). However, we will ignore this potential improvement in our analysis.

class number N	M	mean c_A	min c_A
CSIDH-512	2^{20}	4.406	3.615
$\approx 2^{257}$	2^{16}	5.210	4.307
	2^{12}	6.332	5.187
CSIDH-1024	2^{20}	7.833	6.665
$\approx 2^{512}$	2^{16}	9.882	8.506
	2^{12}	13.553	11.640

Table 1. Experimental results for c_A for different values of M over 10.000 randomly generated lattices in the case of CSIDH-512 as suggested in CSI-SharK. Similarly, experimentally observed values for c_A over 1.000 randomly generated lattices by approximating the class number of CSIDH-1024 by a 512-bit prime.

5.4 Enumeration cost estimate

Clearly, there is a trade-off between the cost of the enumeration to find close vectors in the ℓ_∞ norm and the number of lattices tried. Algorithm 4 has to iterate through

$$\approx (2 \cdot c_A \cdot \frac{M}{N^{1/k}} + 2)^k$$

vectors by Lemma 5.1.

Using amplitude amplification as described in Section 5.1, we can reduce this to $\frac{\pi}{2}(2c_A \cdot \frac{M}{N^{1/k}} + 2)^{k/2}$ steps of amplification. Each step of amplification is dominated by checking if $\|A(x+x_0)-b\|_\infty \leq M$ which requires k^2 multiplications in $\mathbb{Z}_N$. The overall quantum complexity of the enumeration for the concrete parameters at hand thus becomes

$$k^2 \cdot \frac{\pi}{2}(2 \cdot c_A \cdot \frac{M}{N^{1/k}} + 2)^{k/2}$$

times the cost of performing a multiplication in $\mathbb{Z}_N$.

Trading multiplications for additions. Multiplications in $\mathbb{Z}_N$, according to the estimate from Pavlidis and Gizopoulos [52, Table 4], would require a circuit using $2^{9.6}n^2 + 2^{9.3}n$ gates (for $n = \log N$) per multiplication. From Draper [31] the quantum cost of integer addition is around $n \log n$ gates. Thus we can try to change these multiplications to additions to gain some savings. Recall from Algorithm 3, Step 3, that the computation being repeated is to check if

$$\|A(x + x_0) - b\|_\infty \leq M.$$

In each of the iterations, the variables A, x_0, b, M are all fixed so the values of $Ax_0 - b$ can be precomputed. Since $-M \leq x_j \leq M$, we can write $x_j = -M + \sum_{k=0}^{\lceil \log_2 M \rceil + 1} x_{jk} 2^k$ with $x_{jk} \in \{0, 1\}$. We then have $(Ax)_i = -M \sum_j A_{ij} + \sum_{jk} A_{ij} x_{jk} 2^k$ where the values $-M \sum_j A_{ij}$ and $(A_{ij} 2^k \mod N)$ can be precomputed. This requires $k^2 \log M$ *additions*. In total, in each iteration this replaces k^2 multiplications with $k^2 \log M$ additions. This leaves the complexity of the enumeration at

$$\log N \cdot \log \log N \cdot \log M \cdot k^2 \cdot \frac{\pi}{2}(2 \cdot c_A \cdot \frac{M}{N^{1/k}} + 2)^{k/2}$$

quantum T-gates.

We note that memory requirements of the enumeration approach are small, essentially corresponding to the matrix-vector multiplication circuit.

6 Instantiating the attack with sieving

In this section, we explore an alternative approach to solve the CVP instance from Eq. (4) through sieving.

6.1 Sieving approach for CVP problems

Using Kannan's embedding [39, p. 437], our CVP instance translates to a shortest vector problem (SVP) in the ℓ_∞ norm in the lattice H. Then the SVP instance

we would like to solve (in the ℓ_∞ norm) is

$$\begin{bmatrix} i_2 \cdots i_k \ \lambda \ -1 \end{bmatrix} \begin{bmatrix} y_2' & 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ y_k' & 0 & & 1 & \vdots \\ N & 0 & \cdots & 0 & 0 \\ \alpha' & 0 & \cdots & 0 & M \end{bmatrix} = \begin{bmatrix} -i_1 \ i_2 \cdots i_k \ M \end{bmatrix}, \tag{5}$$

where the targeted short vector has infinity norm M.

Sieving algorithms proceed in two phases to solve the short vector problem:

1. *Sampling*, where many elements v_i of the lattice are sampled.
2. *Sieving*, where given a list of lattice elements, another list of elements with smaller norms is produced. In practice, this is done by identifying pairs of "close" vectors and keeping their difference (a smaller vector of the lattice).

The sieving is repeated until we get a vector of the desired norm. To fix some notation, we summarize this general approach in Algorithm 5.

Algorithm 5 General approach of sieving algorithms

Input: A basis B of the lattice L, suitable parameters $(R_0, s_0), ..., (R_{\mathsf{steps}}, s_{\mathsf{steps}})$, a sampling algorithm SAMPLE that given B produces a random element of the lattice L of norm less than R_0 and a sieving algorithm SIEVE that given a list of cardinality s_i with lattice vectors of size R_i produces a list of cardinality s_{i+1} with lattice vectors of size R_{i+1}.

Output: An element of L of norm less than R_{steps}.

```
    # Sampling part
S ← ∅
for i from 1 to s_0
    e_i ← SAMPLE(B, R_0)
    S ← S ∪ {e_i}
end for
    # Sieving part
for i from 1 to steps
    S ← SIEVE(S, R_{i−1}, R_i)
                ▷ SIEVE gets a list of size s_{i−1} and outputs a list of size s_i.
end for
Return a non-zero element of S.
```

To solve the shortest vector problem in the ℓ_∞ norm, we will follow the work of Aggarwal and Mukhopadhyay [1,2]. In Section 6.2 we recall the relevant parts of their approach. We give complexity estimates for the different steps of sampling and sieving depending on the parameters the sieving is instantiated with, i.e. the lattice dimension $k+1$ and the parameters $\{(R_i, s_i)\}$ from Algorithm 5. In Section 6.4, we will describe how these complexity estimates lead to an optimization problem. Finally, we approximate the solution to this optimization problem to obtain an upper bound on the cost of the resulting algorithm.

6.2 Sieving with Aggarwal-Mukhopadhyay's algorithm

Sampling. In [2], a randomized version of Babai's nearest plane algorithm is used to sample elements of the lattice. We are interested in the case where the lattice is of the form as in Eq. (5), i.e. the lattice is defined over $\mathbb{Z}$ but contains $(N\mathbb{Z})^k \times \{0\}$. Therefore, we can sample a random element of the lattice of ℓ_∞ norm less than $N/2$ with $k+1$ multiplications in $\mathbb{Z}_N$, as outlined in the following lemma.

Lemma 6.1. *For lattices of the same shape as H, the procedure (Algorithm 6) of choosing a "small" coefficient for the last row (less than $N/2M$) and choosing random coefficients for the other rows using a representation of the output in $[-N/2, N/2]$ effectively samples a random element of the lattice of norm less than $N/2$.*

Algorithm 6 Sampling algorithm

Input: A matrix $\mathcal{M}$ of the form given 5 defining the lattice L.
Output: An element of L of norm less than $N/2$.

1: Sample a random x_{k+1} in $[-N/2M, N/2M]$ and $x_1, ..., x_{k-1}$ in $[-N/2, N/2]$.
2: Compute $x_k = -\left\lfloor \frac{\alpha' x_{k+1} + \sum_{i=1}^{k-1} y'_{i+1} x_i}{N} \right\rceil$.
3: Return $[x_1, ..., x_{k+1}]\mathcal{M}$.

Proof. The first choice gives that the last coefficient is a random element in $M\mathbb{Z} \cap [-N/2, N/2]$. (The matrix imposes that this coefficient is a multiple of M.) The second choice and the reduction modulo N ensure the randomness of the other coefficients while not affecting the last one. □

Sieving. While [2] explores many refinements for the lattice sieving, these refinements come at a polynomial cost. While asymptotically the exponential cost is what matters most, these added costs are not negligible for the "small" parameters we are interested in. Hence, we only consider the simpler sieving as described in [2, Sect. 3].

Let $\lfloor \cdot \rceil$ denote rounding to the nearest integer. For an element in the lattice $x = (x_1, \ldots, x_{k+1})$, we denote component-wise rounding by $\lfloor 2x/R_i \rceil := (\lfloor 2x_1/R_i \rceil, \ldots, \lfloor 2x_{k+1}/R_i \rceil)$.

Let x and x' be two elements of the lattice such that $\lfloor 2x/R_i \rceil = \lfloor 2x'/R_i \rceil$. Then $x - x'$ must be an element of the lattice with norm smaller than R_i. Therefore, sieving can be done by looking for collisions in the function $x \mapsto \lfloor 2x/R_i \rceil$ (see also [2]).

To estimate the cost of finding such collisions we assume that elements of the lattice behave randomly (as in [2]) and we use the following lemma.

Heuristic 1 *At any stage of the sieving process, the vectors in* $S \cap B_\infty(R_i)$ *are uniformly distributed in* $B_\infty(R_i) = \{x \in \mathbb{Z}^{k+1} | \|x\|_\infty \leq R_i\}$.

Lemma 6.2. *For a random function* $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$*, the number of collisions* $\#\{\{i, j\} : i \neq j, f(i) = f(j)\}$ *has expected value* $\frac{n(n-1)}{2m}$ *and variance of* $\frac{n(n-1)}{2m}\left(1 - \frac{1}{m}\right)$.

The proof of Lemma A.1 is included in Appendix A.

Assume that the map $x \mapsto \lfloor 2x/R_i \rceil$ behaves like a random function on elements of norm less than R_{i-1} in our $k+1$-dimensional lattice, i.e. it maps the elements uniformly randomly to the possible $\left(\frac{2R_{i-1}}{R_i}\right)^{k+1}$ buckets. Starting from a list of 2^n random elements of norms less than R_{i-1} and applying Lemma A.1, we then can get a list of at most $2^{2n-1}\left(\frac{R_i}{2R_{i-1}}\right)^{k+1}$ with standard deviation $2^{n-\frac{1}{2}}\left(\frac{R_i}{2R_{i-1}}\right)^{(k+1)/2}$ elements of norm less than R_i. Note that we ignored the factor $(1 - \frac{1}{m})$ from the variance given by Lemma A.1 since m is large in our application.

This formula will allow us to give constraints on the required suitable values for (R_i, s_i).

Classical "many collisions finding" algorithm. As part of the sieving process, we need an algorithm to compute collisions. Note that there is not just one collision to find in this context, but *many* of them, and that this computation will be performed as part of Step 11 of Algorithm 1, i.e. by a quantum computer on inputs in superposition. We use an algorithm from [20], which essentially implements Algorithm 7 reversibly. Note that this algorithm requires a QRAM containing nearly all elements, but with stricter conditions on the data structure.

Algorithm 7 Classical many collision algorithm

Input: A function $f : \{1, \ldots, 2^n\} \to \{1, \ldots, 2^m\}$.
Output: 2^t collisions.
1: **for** i from 1 to $2^{(t+m+1)/2}$
2: $\quad S_i = \{\}$
3: **end for**
4: **for** i from 1 to $2^{(t+m+1)/2}$
5: $\quad S_{f(i) \bmod 2^{(t+m+1)/2}} = \{(i, f(i))\} \cup S_{f(i) \bmod 2^{(t+m+1)/2}}$
6: **end for** ▷ The elements with the same image by f are now next to each other
7: Walk through the different S_i and list the pairs of elements with the same output by f in S'.
8: Return S'

Lemma 6.3. *Let the notation be as in Algorithm 5. Using classical many collisions finding, there exists a quantum procedure for sieving that takes a list of s_{i-1} elements of norms less than R_{i-1}, and outputs a list of s_i elements of norms less than R_i in time*

$$\frac{1}{\log_2(e)}(\log_2(s_i) + (k+1)\log_2(2R_{i-1}/R_i) + 1)\sqrt{2s_i}\left(\frac{2R_{i-1}}{R_i}\right)^{(k+1)/2}$$

whenever $s_i \leq \frac{s_{i-1}^2}{2}\left(\frac{R_i}{2R_{i-1}}\right)^{k+1}$.

Proof. The classical many collisions finding algorithm (Algorithm 7) takes a function $f : \{1, \ldots, 2^n\} \to \{1, \ldots, 2^m\}$ and returns 2^t collisions. This is an implementation of a hash table. Its complexity is $\frac{1}{\log_2(e)}(t+m+1)2^{(t+m+1)/2}$ bit operations.

We now apply this cost evaluation of the many collision finding algorithm to the function $x \mapsto \lfloor 2x/R_i \rceil$ for sieving (using the notation of Algorithm 5). Recall, the sieving procedure takes a list of s_{i-1} elements of norm less than R_{i-1} and outputs a list of s_i elements of norm less than R_i. Substituting $2^n, 2^m$ and 2^t by s_{i-1}, $\left(\frac{2R_{i-1}}{R_i}\right)^{k+1}$ and s_i, respectively, yields the result of the lemma.

Quantum collision finding We sketch an alternative quantum approach to running the many collisions finding algorithm in Appendix A. We leave the concrete analysis of this approach to further work.

6.3 Returning a solution

We recall Equation 5

$$\begin{bmatrix} i_2 & \cdots & i_k & \lambda & -1 \end{bmatrix} \begin{bmatrix} y_2' & 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ y_k' & 0 & & 1 & \vdots \\ N & 0 & \cdots & 0 & 0 \\ \alpha' & 0 & \cdots & 0 & M \end{bmatrix} = \begin{bmatrix} -i_1 & i_2 & \cdots & i_k & M \end{bmatrix}.$$

At Step 9 in Algorithm 5, we get a list of elements of norm less than M. The last component of the vector is a multiple of M so it can only be one of $\{-M, 0, M\}$.

A vector of the lattice that is of norm less than M and whose last component is M is a solution to the problem. If we get a vector of the lattice that is of norm less than M and whose last component is $-M$, we can just multiply it by -1 to get a solution.

Assuming that the lattice elements behave randomly, we expect that a vector of the list leads to a solution with probability 2/3. So given a list of three solutions to the SVP problem, the probability that all three have last component 0 is

$(1/3)^3 = 1/81$. The complement probability, the case where at least one has a non-zero final component (and is thus usable), is more than 98.7%. By Equation 2, this is large enough to guarantee a large overall success probability.

6.4 Solving the optimization problem

To solve a concrete instance of the SVP using sieving as sketched in Algorithm 5, Lemma 6.3 provides constraints and guidance to set the values for s_i and R_i in the sieving procedure.

To simplify notation, we define $n_i := \log_2(s_i)$ and $m_i := (k+1)\log_2(2R_{i-1}/R_i)$ for this subsection. Further, we recall that steps denotes the number of iterations of the sieving step, which is also a parameter to optimize.

As described before, we have to consider the following optimization problem:

Problem 6.4. Given variables $n_0, \ldots, n_{\mathsf{steps}}$ and $m_1, \ldots, m_{\mathsf{steps}}$ we have the following constraints:

$$\begin{cases} 2^{n_0} \leq \frac{N^{k+1}}{M^2}, \\ n_i \geq 1, \\ n_{i+1} \leq 2n_i - m_{i+1} - 1, \\ \sum_{i=1}^{\mathsf{steps}} (m_i - (k+1)) = (k+1)\log_2(N/M). \end{cases}$$

The complexity of the algorithm is then given by the function

$$(k+1)C_N 2^{n_0} + \sum_{i=1}^{\mathsf{steps}} \frac{2(n_i + m_i + 1)}{\log_2(e)} 2^{(n_i+m_i+1)/2}$$

T-gates, where C_N is the cost of a multiplication in $\mathbb{Z}_N$. We now show how to minimize this cost.

Since N^k is significantly larger than M, the first condition is satisfied for every possible parameter set relevant to us. The other conditions are linear, so the solution set can be treated as a polyhedron. For simplicity we will also replace$n_{i+1} \leq 2n_i - m_{i+1} - 1$ with $n_{i+1} = 2n_i - m_{i+1} - 1$. The cost function to be optimized, however, is not linear. Furthermore, the value steps is not fixed. In order to find good enough solutions we do the following. We fix steps to an appropriate value and then treat this as a linear program. The objective function to be optimized is a bit trickier as the coefficient of n_0 is much larger in our applications as the other variables. Hence we choose the objective function as $C_n \cdot n_0 + \sum_{i=1}^{n} n_i$ (note that the n_i already define the m_i). This alone would yield horrible results as optimal solutions are not balanced and one large n_i is enough to make the complexity explode. By imposing an upper bound on all n_i depending on the dimension k, we can prevent this. Once a solution is computed,

we plug it into the function $(k+1)C_N 2^{n_0} + \sum_{i=1}^{\mathsf{steps}} \frac{2(n_i+m_i+1)}{\log_2(e)} 2^{(n_i+m_i+1)/2}$ to get the complexity of sieving.

What we observe experimentally is that there seems to be an optimal choice for the upper bound on the n_i and the value steps. It makes sense to take steps small, albeit large enough that a solution exists.

We choose $\mathsf{steps} = 600$ and the upper bound on the n_i to be $k+3$. We do not prove that this is an optimal choice (though experimentally we found this to be the case) as we only care about the resulting complexity. The following is a table which lists the number of operations and QRAM necessary for various k values after computing solutions to the optimization problem[8].

k	13	16	20	24	29	31	40
complexity	$2^{33.0}$	$2^{35.0}$	$2^{38.9}$	$2^{43.2}$	$2^{48.4}$	$2^{50.5}$	$2^{59.8}$

Table 2. We list the resulting complexity in number of T-gates and QRAM for running the sieving approach to solving the knapsack problem for various k values.

7 Quantum security of the Vectorization with Shifted Inputs

We now apply our analysis to assess the security of the Vectorization Problem with Shifted Inputs, in particular for CSIDH parameters as suggested in the CSI-SharK and BCP protocols.

Recall that the concrete complexity of Childs-van Dam's algorithm is

$$(3k+1)\ \text{QFTs}\ + k\ \text{group action evaluations}\ +\ \text{knapsack problem}.$$

Currently the cost of one group action evaluation is estimated to be between 2^{40} [11] and $2^{52.4}$ [21] T-gates. We denote this cost by G. The cost of a QFT is estimated to be around $n(n+1)/2$, where $n = \log N$.

For the enumeration approach, we estimated that the total cost of solving the knapsack problem is

$$L_{enum}(c_A) := n \cdot \log n \cdot \log M \cdot k^2 \cdot \frac{\pi}{2}(2 \cdot c_A \cdot \frac{M}{N^{1/k}} + 2)^{k/2}$$

quantum T-gates, where suitable values for c_A are listed in Table 1. If we choose to resample the lattice around 2^{10} times, we could use the smaller c_A values from Table 1 at the cost of more group action calls and QFTs. Without this

[8] The optimization was run with the Magma computer algebra system [22] and the code is provided at https://anonymous.4open.science/r/CvD-analysis-A4E7/.

resampling, we must use the average c_A values listed in the same table. As a first approximation [9], the total cost of running Childs-van Dam's algorithm will then be bounded by the minimum between

$$(3 \cdot 2^{10} \cdot k + 1)\frac{n(n+1)}{2} + 2^{10}k \cdot G + L_{enum}(\min\ c_A)$$

and

$$(3k + 1)\frac{n(n+1)}{2} + k \cdot G + L_{enum}(\text{mean}\ c_A)$$

T-gates, where G is the cost of group action evaluation.

For the sieving approach, the complexity of solving the knapsack problem, $L_{sieving}$, was computed experimentally in Table 2. The final complexity will thus be

$$(3k + 1)\frac{n(n+1)}{2} + k \cdot G + L_{sieving}$$

T-gates.

In Table 3, we list the concrete total quantum security complexities of the Childs-van Dam algorithm depending on the choice of CVP solver, and on the choice of cost model for the oracle query. Note that $M = 2^{12}$ was suggested as a valid parameter in CSI-SharK, and $M = 2^{12}, 2^{15}, 2^{18}$ were suggested in BCP. This means that we improve upon the state-of-the-art complexity estimate for both CSI-SharK and BCP. Overall, we were able to decrease the number of oracle calls required (compared to running Kuperberg on a single shift) to solve the Vectorization Problem with Shifted Inputs.

Childs-van Dam vs Kuperberg As the parameters k and M in Childs-van Dam's algorithm are related by $k := \log N/(\log(M) + 1)$, a larger M value leads to a smaller k value, i.e. a smaller-dimensional lattice and an easier CVP instance. For the enumeration approach, on top of a smaller dimension, we also (experimentally) get a smaller minimum value of the constant c_A.

When very few shifts are available, Peikert's version of Kuperberg still performs better than our version of the Childs-van Dam algorithm, due to the various overheads in the latter. As the number of available shifts increases, Childs-van Dam becomes the most efficient algorithm. From Table 3 the cross-over point is below 2^8, so as mentioned above the parameters suggested by CSI-SharK are affected.

We note that the sieving approach consistently performs better than the enumeration approach in T-gate complexity. On the other hand, we recall that the enumeration approach can be performed with low memory complexity, hence it could also be deemed more practical depending on the quantum cost metric adopted. In both cases, group action evaluation queries eventually become the dominating costs. At that point, increasing M further only marginally decreases the T-gate complexity.

[9] A slightly better bound can in theory be obtained by considering more options for the number of resampling and the resulting c_A values, but for the parameters we considered this only made very small difference.

group action cost estimate	Peikert Alg. [53]	M	Alg. 1 + sieving	Alg. 1 + enumeration
2^{40} [11]	2^{54} to 2^{59}	2^{8}	$2^{48.4}$	$2^{73.9}$
		2^{12}	$2^{44.3}$	$2^{54.3}$
		2^{16}	$2^{44.0}$	$2^{52.7}$
		2^{20}	$2^{43.7}$	$2^{47.3}$
$2^{52.4}$ [21]	$2^{66.4}$ to $2^{71.4}$	2^{8}	$2^{57.3}$	$2^{73.9}$
		2^{12}	$2^{56.7}$	$2^{56.7}$
		2^{16}	$2^{56.4}$	$2^{56.4}$
		2^{20}	$2^{56.1}$	$2^{56.1}$

Table 3. Using the class group from the CSIDH-512 parameter set, we list the complexity in number of T-gates of the Childs-van Dam algorithm when given access to shifts of the form $g^{c \cdot z}$ for $c \in [-M, M]$. We compare against the algorithm from Peikert, based on Kuperberg's algorithm (which uses between 2^{14} and 2^{19} oracle calls depending on the length of the phase vector). Note that for $M = 2^8$, we ran a partial search for the constant c_A, and estimate $\min c_A \approx 7.7$ and mean $c_A \approx 9.2$.

group action cost estimate	Peikert Alg. [53]	M	Alg. 1 + sieving	Alg. 1 + enumeration
$2^{57.6}$ [21]	$2^{78.4}$ to $2^{85.5}$	2^{12}	$2^{62.9}$	$2^{105.8}$
		2^{16}	$2^{62.6}$	$2^{85.6}$
		2^{20}	$2^{62.2}$	$2^{72.2}$

Table 4. Using the group action estimates from Bonnetain and Schrottenloher for the CSIDH-1024 parameters, we compare the CvD algorithm variants to Peikert's algorithm (based off of Kuperberg) in number of T-gates.

CSIDH-1024 parameters Bonnetain and Schrottenloher [21, Table 3], as well as Peikert [53, Figure 1] also give complexity estimates for the CSIDH-1024 parameter set. Using these analyses, we compare how our version of Childs-van Dam's algorithm compares to Peikert's algorithm in Table 4 with access to M shifts.

8 Conclusion and Perspectives

In this work, we revisited the concrete quantum security of the Vectorization Problem with Shifted Inputs, a variant of the vectorization problem used in isogeny-based cryptography. The best quantum cryptanalysis attacks on the Vectorization Problem with Shifted Inputs consisted in attacking the Vectorization Problem itself, and parameter selection in [7, 8] suggests that the two problems are considered equivalently hard to solve in practice.

In contrast, our work suggests that the two problems are in fact not equivalent with respect to quantum computers. By specifying and analyzing a quantum algorithm of Childs and van Dam, we leverage the additional information provided in the Vectorization Problem with Shifted Inputs, and we obtain new quantum attacks with lower T-gate complexities. A core subtask in Childs-van Dam's algorithm consists in solving a knapsack problem. We considered two approaches to solve it based on enumeration and sieving strategies. Our analysis suggests that the sieving approach has better T-gate complexity for relevant parameters, while the enumeration stands out for its low memory requirements.

Potential improvements In this work we did not consider using Voronoi sets to solve the instance of CVP due to the high memory requirements, however, this approach may offer benefits of its own. When working with ℓ_p norms for $p \neq 2$, the Voronoi set can be superexponential [14], but it is not clear if this is the case in the instance of CSI-SharK. A second area that may benefit from independent study is the parameter selection for the sieving approach in Section 6.4. Here, the parameters $\{n_i, m_i\}$ had to be selected in such a way that optimized the final complexity of the algorithm. More sophisticated optimization algorithms could lead to improved results.

Countermeasures A potential countermeasure for CSI-SharK may be to choose the shifts such that they are not consecutive integers. Doing so may not be trivial, however, as variants of this attack would still apply in some instances. Using a punctured list of consecutive integers (i.e. simply skipping a few values) does not completely avoid the attack, but only decreases the probability of success depending on how many holes there are. If the set of integers is uniformly spaced, for example, they are of the form $\{2, 4, 6, \ldots\}$, then we can replace $f(b, x)$ from Section 3 by $g(b, x) = f(2b, x)$ and the period becomes $(1, 2z)$ instead of $(1, z)$. Note that increasing the size of the integers appearing in the (super)exceptional sets would in turn allow to decrease k. Otherwise, the rest of the attack follows just the same.

Related protocols In this work, we focused on the Vectorization Problem with Shifted Inputs, a variant of vectorisation problem used in the CSI-SharK and BCP protocols. Many other variants have appeared in the literature, some of them partly similar to the same problem. We list some of these problems below, and we encourage the community to extend our results and study their exact quantum security.

CSI-Otter. In [40], the authors give a (partially) blind signature scheme from isogenies called CSI-Otter. Though the soundness of the scheme was recently attacked in a special case [41], the scheme in practice is unaffected. In particular, its underlying hard problem, ζ_d-rGAIP [40, Definition 2.9], remains unaffected. We state it here.

Problem 8.1 (ζ_d-Ring Group Action Inverse Problem). Let $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$. Given

$$E \cup \{[\mathfrak{g}^{\zeta_d^j \cdot z}]E\}_{j\in[d]},$$

where $z \in \mathbb{Z}_N$ is secret and $d|\lambda(N)$ (here λ is the Carmichael function), compute z.

This problem is very similar to that of Problem 2.1 except that the shifts are not uniformly spaced. Can adjustments be made to Algorithm 1 to accommodate this difference?

k-power DDHA. Another hard problem to consider is the *k-power Decisional Diffie-Hellman Group Action Problem* which was first used in an isogeny-based threshold signature by De Feo and Meyer [34, Problem 1] but was later generalized to any group action in [32, Definition 8]. This problem (when $k = 2$) was also used in a quantum money construction from Zhandry [56], who suggests isogenies may be a good candidate for instantiation. We state the problem in terms of the isogeny group action.

Problem 8.2 (k-power Decisional Diffie-Hellman Group Action Problem). Let $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$, $1 < k < N$ an integer, and $\mathfrak{g}^z \in \mathrm{cl}(\mathcal{O})$. Given $(k, E, [\mathfrak{g}^z]E, F)$, where $F \in \mathcal{E}\ell\ell_p(\mathcal{O})$, determine whether $F = [\mathfrak{g}^{k\cdot z}]E$ or whether it was sampled uniformly randomly from $\mathcal{E}\ell\ell_p(\mathcal{O})$.

At first sight, when F is not random, this problem could be viewed as a hidden shift problem where we have access to two shifts, namely $(E, [\mathfrak{g}^z]E, [\mathfrak{g}^{k\cdot z}]E)$. Thus we could attempt to run the algorithm using $(E, [\mathfrak{g}^z]E, F)$. If we obtain a viable candidate for z, then F was in fact of the form $F = [\mathfrak{g}^{k\cdot z}]E$, otherwise it was random. The main issues here are that due to the very small number of shifts, and the possibility of k being very large, the probability of the algorithm returning a correct answer when F is not random will not be very high. Nonetheless, this approach may shed some light on the quantum security of [34], either affirming its claims or improving the state-of-the-art.

An OPRF from CSIDH. In [29], Delpech de Saint Guilhem and Pedersen give an OPRF from CSIDH. In their proof of one-more unpredictability, the adversary has access to an OPRF oracle that on input $m \in \mathbb{Z}_N$ outputs $[\mathfrak{g}^{f(m)}]E$, where f is a secret polynomial of the form $f(x) = a(x+b)^3 + c$. The adversary in this context can make polynomially many queries to the oracle, however, due to the fact that the polynomial is kept secret, it is not possible to know what the shifts are.

Other group action based cryptography. None of our work relies on the use of isogenies, and so applying this attack to other abelian group actions remains an interesting question.

References

1. Divesh Aggarwal and Priyanka Mukhopadhyay. Improved algorithms for the shortest vector problem and the closest vector problem in the infinity norm. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 35:1–35:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
2. Divesh Aggarwal and Priyanka Mukhopadhyay. Improved algorithms for the shortest vector problem and the closest vector problem in the infinity norm. *arXiv preprint arXiv:1801.02358*, 2018.
3. Graeme Ahokas, Richard Cleve, and Lisa Hales. The complexity of quantum fourier transforms and integer multiplication. ERATO Conference on Quantum Information Science, 2003.
4. Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 601–610. ACM, 2001.
5. Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439. Springer, 2020.
6. Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.
7. Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. CSI-SharK: CSI-FiSh with sharing-friendly keys. Cryptology ePrint Archive, Paper 2022/1189, 2022. https://eprint.iacr.org/2022/1189.
8. Karim Baghery, Daniele Cozzo, and Robi Pedersen. An isogeny-based ID protocol using structured public keys. In Maura B. Paterson, editor, *Cryptography and Coding - 18th IMA International Conference, IMACC 2021, Virtual Event, December 14-15, 2021, Proceedings*, volume 13129 of *Lecture Notes in Computer Science*, pages 179–197. Springer, 2021.
9. Naomi Benger, Dario Catalano, Manuel Charlemagne, David Conti, Biljana Cubaleska, Hernando Fernando, Dario Fiore, Steven Galbraith, David Galindo, Jens Hermans, Vincenzo Iovino, Tibor Jager, Markulf Kohlweiss, Benoit Libert, Richard Lindner, Hans Loehr, Danny Lynch, Richard Moloney, Khaled Ouafi, Benny Pinkas, Frantisek Polach, Mario Di Raimondo, Markus Ruckert, Michael Schneider, Vijay Singh, Nigel Smart, Martijn Stam, Fre Vercauteren, Jorge Villar Santos, and Steve Williams. Main computational assumptions in cryptography. *European Network of Excellence in Cryptology II*, 2010.
10. Charles H Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17:525–532, 1973.
11. Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 409–441. Springer, 2019.

12. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. *Advances in Cryptology – ASIACRYPT 2019*, pages 227–247, 2019.
13. Jean-François Biasse, David Jao, and Anirudh Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology – INDOCRYPT 2014*, pages 428–442, Cham, 2014. Springer International Publishing.
14. Johannes Blömer and Kathlén Kohn. Voronoi cells of lattices with respect to arbitrary norms. *SIAM J. Appl. Algebra Geom.*, 2(2):314–338, 2018.
15. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.
16. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.
17. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.
18. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
19. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.
20. Xavier Bonnetain, André Chailloux, André Schrottenloher, and Yixin Shen. Finding many collisions via reusable quantum walks: Application to lattice sieving. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 221–251. Springer, 2023.
21. Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 493–522. Springer, 2020.
22. Wieb Bosma, John Cannon, and Catherine Playoust. The magma algebra system i: The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.
23. Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

24. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.
25. Jung Hee Cheon. Discrete logarithm problems with auxiliary inputs. *J. Cryptol.*, 23(3):457–476, 2010.
26. Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *J. Mathematical Crytology*, 8(1):1–29, 2014.
27. Andrew M Childs and Wim Van Dam. Quantum algorithm for a generalized hidden shift problem. *arXiv preprint quant-ph/0507190*, 2005.
28. David A. Cox. *Class Field Theory*, chapter 2, pages 87–179. John Wiley & Sons, Ltd, 2013.
29. Cyprien Delpech de Saint Guilhem and Robi Pedersen. New proof systems and an OPRF from CSIDH. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part III*, volume 14603 of *Lecture Notes in Computer Science*, pages 217–251. Springer, 2024.
30. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer, 2005.
31. Thomas G. Draper. Addition on a quantum computer, 2000.
32. Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part I*. Springer, 2023.
33. Friedrich Eisenbrand, Nicolai Hähnle, and Martin Niemeier. Covering cubes and the closest vector problem. In Ferran Hurtado and Marc J. van Kreveld, editors, *Proceedings of the 27th ACM Symposium on Computational Geometry, Paris, France, June 13-15, 2011*, pages 417–423. ACM, 2011.
34. Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 187–212. Springer, 2020.
35. Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 2007–2023. ACM, 2020.
36. Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

37. Jr. H. W. Lenstra. Integer programming with a fixed number of variables. Mathematics of Operations Research, Vol. 8, No. 4 (Nov., 1983), pp. 538-548, 1983.
38. Guillaume Hanrot and Damien Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 170–186, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
39. Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.
40. Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI -otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 729–761. Springer, 2023.
41. Shuichi Katsumata, Yi-Fu Lai, and Michael Reichle. Breaking parallel ROS: implication for isogeny and lattice-based blind signatures. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part I*, volume 14601 of *Lecture Notes in Computer Science*, pages 319–351. Springer, 2024.
42. Taechan Kim. Security analysis of group action inverse problem with auxiliary inputs with application to CSIDH parameters. In Jae Hong Seo, editor, *Information Security and Cryptology - ICISC 2019 - 22nd International Conference, Seoul, South Korea, December 4-6, 2019, Revised Selected Papers*, volume 11975 of *Lecture Notes in Computer Science*, pages 165–174. Springer, 2019.
43. Neal Koblitz and Alfred Menezes. Critical perspectives on provable security: Fifteen years of "another look" papers. *Adv. Math. Commun.*, 13(4):517–558, 2019.
44. Stephan Krenn, Omid Mir, and Daniel Slamanig. Structure-preserving compressing primitives: Vector commitments, accumulators and applications. Cryptology ePrint Archive, Paper 2024/1619, 2024.
45. Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.
46. Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In Simone Severini and Fernando G. S. L. Brandão, editors, *8th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2013, May 21-23, 2013, Guelph, Canada*, volume 22 of *LIPIcs*, pages 20–34. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.
47. László Lovász and Herbert E. Scarf. The generalized basis reduction algorithm. *Math. Oper. Res.*, 17(3):751–764, 1992.
48. Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 575–584, 2007.
49. Olivia Di Matteo, Vlad Gheorghiu, and Michele Mosca. Fault-tolerant resource estimation of quantum random-access memories. *IEEE Transactions on Quantum Engineering*, 1:1–13, 2020.
50. Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *SIAM J. Comput.*, 42(3):1364–1391, 2013.
51. Aurel Page and Damien Robert. Introducing clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Paper 2023/1766, 2023.

52. Archimedes Pavlidis and Dimitris Gizopoulos. Fast quantum modular exponentiation architecture for shor's factoring algorithm. *Quantum Inf. Comput.*, 14(7-8):649–682, 2014.
53. Chris Peikert. He gives c-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 463–492. Springer, 2020.
54. Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space, 2004.
55. Charles Yuan and Michael Carbin. Tower: data structures in quantum superposition. *Proceedings of the ACM on Programming Languages*, 6(OOPSLA2):259–288, 2022.
56. Mark Zhandry. Quantum money from abelian group actions. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 101:1–101:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

A More on Collision Finding

Lemma A.1. *For a random function $f : \{1, \dots, n\} \to \{1, \dots, m\}$, the number of collisions $\#\{\{i, j\} : i \neq j, f(i) = f(j)\}$ has expected value $\frac{n(n-1)}{2m}$ and variance of $\frac{n(n-1)}{2m}\left(1 - \frac{1}{m}\right)$.*

Proof. Let $X_i = \#\{x : f(x) = i\}$. For any $a_1, \dots, a_m \in \mathbb{N}$ such that $\sum_i a_i = n$, we have

$$\Pr_f(X_1 = a_1, \dots, X_m = a_m) = \frac{n!}{a_1! \dots a_m! m^n}$$

as $\Pr_f(X_1 = a_1, \dots, X_m = a_m) =$ (number of choices for X_1)$\times$(number of remaining choices for X_2)$\times \dots$ (number of remaining choices for X_m) divided by all choices. This can be written out as

$$\frac{\binom{n}{a_1}\binom{n-a_1}{a_2} \dots \binom{n-a_1-a_2-\dots-a_{m-1}}{a_{m-1}}}{m^n} = \frac{n!}{a_1! \dots a_m! m^n}.$$

Now let us compute $E_f[x_i]$ which is clearly the same for every i as the X_i are identically distributed. Observe that $E_f[\sum_{i=1}^m X_i] = n$ by definition. Since expectation is linear this implies that $E_f[X_i] = \frac{n}{m}$.

The following equations follow from the fact that X_i follow a multinomial distribution of parameters n and $(1/m, ..., 1/m)$.

$$\mathbb{E}_f[X_i] = \frac{n}{m},$$
$$\mathbb{E}_f[X_i(X_i - 1)] = \mathbb{E}_f[X_i X_j] = \frac{n(n-1)}{m^2},$$

$$\begin{aligned}\mathbb{E}_f[X_i(X_i-1)(X_i-2)] &= \frac{n(n-1)(n-2)}{m^3},\\ \mathbb{E}_f[X_i(X_i-1)(X_i-2)(X_i-3)] &= \mathbb{E}_f[X_i(X_i-1)X_j(X_j-1)]\\ &= \frac{n(n-1)(n-2)(n-3)}{m^4}.\end{aligned}$$

Observe that

$$\#\{(i,j) : i \neq j, f(i) = f(j) = k\} = \frac{X_k(X_k-1)}{2}$$

as X_k elements are mapped to k and then any two of them provide a collision. This implies that

$$\#\{\{i,j\} : i \neq j, f(i) = f(j)\} = \sum_i \frac{X_i(X_i-1)}{2}.$$

Since expectation is linear and $\mathbb{E}_f[X_i(X_i-1)] = \frac{n(n-1)}{m^2}$ we get that the expectation of the collisions is $\frac{1}{2}m\frac{n(n-1)}{m^2} = \frac{n(n-1)}{2m}$. The result on variance follows from the previous equations and $V(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

$$\begin{aligned}\mathbb{E}_f[(\#collisions)^2] &= \mathbb{E}_f[\left(\sum_i X_i(X_i-1)/2\right)^2]\\ &= 1/4\sum_{i,j}\mathbb{E}_f[X_i(X_i-1)X_j(X_j-1)]\\ &= 1/4\sum_{i\neq j}\mathbb{E}_f[X_i(X_i-1)X_j(X_j-1)] + 1/4\sum_{i=j}\mathbb{E}_f[X_i(X_i-1)X_j(X_j-1)]\\ &= \frac{n(n-1)(n-2)(n-3)m(m-1)}{4m^4} + 1/4\sum_i \mathbb{E}_f[X_i(X_i-1)X_i(X_i-1)]\\ &= \frac{n(n-1)(n-2)(n-3)m(m-1)}{4m^4} + 1/4\sum_i \mathbb{E}_f[X_i(X_i-1)(X_i-2)(X_i-3)\\ &\qquad +4X_i(X_i-1)(X_i-2) + 2X_i(X_i-1)]\\ &= \frac{n(n-1)(n-2)(n-3)(m-1)}{4m^3} + \frac{n(n-1)(n-2)(n-3)}{4m^3}\\ &\qquad + \frac{n(n-1)(n-2)}{m^2} + \frac{n(n-1)}{2m}\\ &= \frac{n(n-1)(n-2)(n+1)}{4m^2} + \frac{n(n-1)}{2m}\\ V(\#collisions) &= \frac{n(n-1)(n-2)(n+1)}{4m^2} + \frac{n(n-1)}{2m} - \frac{n(n-1)n(n-1)}{4m^2}\\ &= \frac{n(n-1)}{2m}\left(1-\frac{1}{m}\right)\square\end{aligned}$$

A.1 Quantum many collision finding algorithm

We analyse the non-asymptotic complexity of the quantum algorithm for many collisions finding given by Bonnetain, Chailloux, Schrottenloher and Shen [20]. This algorithm is based on the analysis of quantum walks from Magniez, Nayak, Roland and Santha [48] and the use of quantum walks for collisions from Ambainis [6].

Quantum walks The quantum walk algorithm is quantum search analogous to Grover's algorithm [36] where the set of the search is a regular graph. For further use, we note the proportion of desired vertices ϵ and the spectral gap of the graph δ.

Setup cost S**.** The cost of producing the uniform superposition of vertices.
Update cost U**.** The cost of making the uniform superposition of neighbors of a vertex.
Checking cost C**.** The cost of checking if a vertex is a desired one.

From [48], for an integer s, the total cost of a quantum walk is less than (but close to)

$$S + \frac{\pi}{2\sqrt{\epsilon}}\left(\left(\frac{8\pi}{\sqrt{\delta}}U + \frac{1}{2}\left(\log\frac{2\pi}{\sqrt{\delta}} + s\right)^2 + \frac{3}{2}\left(\log\frac{2\pi}{\sqrt{\delta}} + s\right)\right) + C\right)$$

with success probability more than $(1 - \arcsin\sqrt{\epsilon}) \times \left(1 - \frac{\pi}{2\sqrt{\epsilon}2^s}\right)$.

Quantum walks for collisions. From now on, we consider a function $f : \{0,1\}^n \to \{0,1\}^m$ on which we want to get collisions. The expected number of collisions is roughly 2^{2n-m-1}.

In [6], quantum walks are used for computing collisions. The idea is to apply a quantum walk on a Johnson graph. Let $n, K \in \mathbb{N}$. The Johnson graph $J(2^n, K)$ is the graph of subsets of size K in $\{1, \ldots, 2^n\}$ and two subsets are connected if and only if their intersection is of size $K - 1$. The spectral gap of $J(2^n, K)$ is $\delta = \frac{2^n}{K(2^n-K)}$ The desired vertices are the ones containing a collision, so $\epsilon \simeq \frac{K^2}{2^{m+1}}$.

The Setup, Update and Checking costs depend on the data structure used for the subsets. We use the quantum radix tree structure from [55] for practical estimations. Insertion costs $1440k^2 + 5056k$ gates, and checking if an element is in the tree costs $784k^2 + 1612k + 1$ gates, where k is the height of the tree. Checking is made with the update by checking whether the new element added to the set and the one deleted from the set make a collision or not. For $s = m - 4t + 21$, the cost is (heavily) dominated by the term

$$(1440m^2 + 5056m)K + \frac{2^{m/2}}{\sqrt{K}}(248336m^2 + 744560m)$$

with success probability more than $\left(1 - \frac{K\sqrt{8}}{2^{m/2}}\right)\left(1 - \frac{\pi 2^{m/2}}{K\sqrt{22}^s}\right)$.

Reusable quantum walks for collisions. In [20], a new operation Extract is introduced. The observation is that from the result of the quantum walk, one can extract the collision and the rest can be used as a new setup state for a quantum walk. For 2^t collisions, the total cost is (heavily) dominated by the term

$$(1440m^2 + 5056m)K + \frac{2^{t+m/2}}{\sqrt{K}}(248336m^2 + 744560m)$$

with individual success probability more than $\left(1 - \frac{K\sqrt{8}}{2^{m/2}}\right)\left(1 - \frac{\pi 2^{m/2}}{K\sqrt{2}2^s}\right)$. The second term is not dominant from $s = m - 4t + 21$.

This is optimized for

$$K = \left(\frac{2^{t+m/2+1/2}(15521m + 46535)}{90m + 316}\right)^{2/3},$$

giving a total complexity of

$$\frac{3}{2}2^{2t/3+m/3}\left(248336m^2 + 744560m\right)^{2/3}\left(1440m^2 + 5056m\right)^{1/3} \simeq 2^{2t/3+m/3+16.03}(m^2 + 3.17m)$$

with the condition that $K^2/2 \leq 2^m$ which can be approximated to $t \leq m/4 - 8.18$ and an individual success probability higher than $1/2$ (each pair of the output has probability higher than $1/2$ to be a valid collision).

One can ensure the returned elements to be different by adding a verification to the checking cost. This additional cost is negligible.