A Simpler and More Efficient Reduction of DLog to CDH for Abelian Group Actions

Steven Galbraith[1], Yi-Fu Lai[1,2(✉)], and Hart Montgomery[3]

[1] University of Auckland, Auckland, New Zealand
s.galbraith@auckland.ac.nz
[2] Ruhr-University Bochum, Bochum, Germany
Yi-Fu.Lai@ruhr-uni-bochum.de
[3] Linux Foundation, San Francisco, USA

Abstract. Abelian group actions appear in several areas of cryptography, especially isogeny-based post-quantum cryptography. A natural problem is to relate the analogues of the computational Diffie-Hellman (CDH) and discrete logarithm (DLog) problems for abelian group actions. Galbraith, Panny, Smith and Vercauteren (Mathematical Cryptology '21) gave a quantum reduction of DLog to CDH, assuming a CDH oracle with perfect correctness. Montgomery and Zhandry (Asiacrypt '22, best paper award) showed how to convert an unreliable CDH oracle into one that is correct with overwhelming probability. However, while a theoretical breakthrough, their reduction is quite inefficient: if the CDH oracle is correct with probability ϵ then their algorithm to amplify the success requires on the order of $1/\epsilon^{21}$ calls to the CDH oracle.

We revisit this line of work and give a much simpler and tighter algorithm. Our method only takes on the order of $1/\epsilon^4$ CDH oracle calls and is conceptually simpler than the Montgomery-Zhandry reduction. Our algorithm is also fully black-box, whereas the Montgomery-Zhandry algorithm is slightly non-black-box. Our main tool is a thresholding technique that replaces the comparison of distributions in Montgomery-Zhandry with testing equality of thresholded sets.

1 Introduction

Abelian group actions appear in several areas of cryptography. In isogeny-based post-quantum cryptography there have been several new instantiations of group actions, such as CSIDH [CLM+18], CSI-FiSh [BKV19,EKP20], and SCALLOP [FFK+23]. Isogeny-based group actions have been proven to be versatile in many applications, including but not limited to signature schemes [BKV19,EKP20,DG19], UC-secure oblivious transfer protocols [LGD21,BMM+23], threshold signatures [DM20], (linkable/accountable) ring and group signatures [BKP20,BDK+22], blind signatures [KLLQ23], and PAKE [AEK+22].

A natural problem is to relate the analogues of the computational Diffie-Hellman (CDH) and discrete logarithm (DLog) problems for abelian group actions, which we abbreviate GA-CDH and GA-DLog, respectively, since key

Q. Tang and V. Teague (Eds.): PKC 2024, LNCS 14603, pp. 36–60, 2024.
https://doi.org/10.1007/978-3-031-57725-3_2

exchange protocols based on these instantiations are build upon the group action CDH assumption (GA-CDH) but the underlying group action DLog assumptions (GA-DLog) are much better studied and understood from a hardness perspective. Galbraith, Panny, Smith and Vercauteren [GPSV21] gave a quantum reduction of GA-DLog to GA-CDH, assuming a GA-CDH oracle with *perfect correctness*. Subsequently, Montgomery and Zhandry [MZ22] devised a novel approach and a series of sophisticated procedures to transform an unreliable GA-CDH oracle into one that is correct with an overwhelming probability and showed how to use this in the [GPSV21] framework to build a full quantum reduction of GA-DLog to GA-CDH.

However, [MZ22] is quite inefficient: more precisely, if a GA-CDH oracle is correct with probability ϵ, then the Montgomery-Zhandry algorithm to amplify the success probability to an exponentially low amount, which is necessary for their reduction from GA-CDH to GA-DLog to work, takes on the order of $1/\epsilon^{21}$ calls to the original GA-CDH oracle. To put this into perspective, given a GA-CDH oracle with success rate $1/8$, it requires at least 2^{63} oracle calls in [MZ22] to obtain a GA-CDH a GA-DLog solver with an overwhelming advantage. While [MZ22] was a theoretical breakthrough, its inefficiency means it can only have an extremely limited effect on practical parameter setting for group actions or isogenies.

This brings us to the primary objective of this work:

> *Can we have tighter and simpler approaches for the reduction between* CDH *and DLog and to amplify a* CDH *circuit for abelian group actions?*

We answer this question in the affirmative. We show a GA-CDH to GA-DLog reduction that only requires on the order of $1/\epsilon^4$ calls to a GA-CDH oracle that succeeds with probability ϵ. Moreover, our techniques are considerably simpler than the heavy mathematical machinery used in [MZ22], providing a much more understandable reduction as well.

1.1 Group Actions and Computational Problems

Let G be an abelian group acting transitively on set $\mathcal{X}$ via the operation $\star$. We denote group actions as tuples $(G, \mathcal{X}, \star)$. In this paper we assume this is an effective group action (EGA), meaning that there is an efficient algorithm to compute $g \star x$ for any $g \in G$ and $x \in \mathcal{X}$. The isogeny-based primitive CSIDH [CLM+18], which is one of the main motivations for this work, was originally not known to be an effective group action, but recent work by Page and Robert [PR23] gives a solution to this problem, and in the context of our work this issue can be bypassed using a technique of Wesolowski [Wes22]. We note there are seemingly hard barriers to making a GA-DLog to GA-CDH reduction work for non-EGAs (*restricted* effective group actions), and we refer interested readers to [MZ22] for this, where there is an extensive discussion on the topic.

We now introduce the two main computational problems that arise in (abelian) group action cryptography. The discrete logarithm problem is also known as *vectorization*, and the computational Diffie-Hellman problem as *parallelization*.

GA-DLog: Given $(x, g \star x) \in \mathcal{X}^2$, compute g.
GA-CDH: Given $(x, a \star x, b \star x) \in \mathcal{X}^3$, compute $(ab) \star x$.

Galbraith, Panny, Smith and Vercauteren [GPSV21] showed a quantum reduction of GA-DLog to GA-CDH (vectorization to parallelization) for *perfect* adversaries. At a high level the idea is the following: let $\mathcal{A}$ be a (quantum) oracle such that $\mathcal{A}(a \star x, b \star x) = (ab) \star x$ with overwhelming correctness. Given a GA-DLog instance $(x, a \star x)$ one can use the oracle as $\mathcal{A}(a^m \star x, a^n \star x) = (a^{m+n}) \star x$ to compute $(a^t) \star x$ for any desired $t \in \mathsf{Z}$. Suppose for simplicity that G is cyclic and let g be a generator of G. Define $f : \mathsf{Z}^2 \to \mathcal{X}$ by $f(s,t) = (g^s) \star (a^t) \star x$. One can compute f using the oracle $\mathcal{A}$. Applying Shor's algorithm [Sho94] for the hidden subgroup problem returns an element in the lattice $L = \{(s,t) \in \mathsf{Z}^2 : g^s a^t = 1\}$. If $\gcd(t, |G|) = 1$ then we can solve the discrete logarithm of a to the base g, and hence can compute a.

The intuition of [GPSV21] is that the ability to compute GA-CDH allows us to turn a group action into a group, since we can "multiply" elements using the GA-CDH oracle. This means we can directly apply Shor's algorithm for solving discrete log on groups.

1.2 The Montgomery-Zhandry Approach

Montgomery and Zhandry [MZ22] showed how to handle an oracle that is only correct with probability ϵ. Since the decisional Diffie-Hellman problem for group actions is hard, and since we lack the algebraic tools used in to resolve this problem in the case of CDH in groups, there seems to be no easy way to determine whether or not an output of the oracle is correct or not. We sketch some of the main ideas of their work.

For $y, z \in \mathcal{X}$ define $\mathcal{A}_0(y,z)$ to be the algorithm that samples uniformly at random group elements $a', b' \in G$, and returns $(a'b')^{-1} \star \mathcal{A}(a' \star y, b' \star z)$. Let $\mathcal{D}$ be the output distribution of $\mathcal{A}_0(x,x)$. Montgomery and Zhandry show that $\mathsf{Pr}[x \leftarrow \mathcal{D}] = \epsilon$ and that the output distribution of $\mathcal{A}_0(a \star x, b \star x)$ is the same as the shift of the distribution $\mathcal{D}$, which we denote as $(ab) \star \mathcal{D}$ (meaning $\mathsf{Pr}[w \leftarrow (ab) \star \mathcal{D}] := \mathsf{Pr}[((ab)^{-1} \star w) \leftarrow \mathcal{D}]$). We also use $\mathcal{A}_0$ in this work, as it is a very basic self-reduction. We explain $\mathcal{A}_0$ in more detail in the body of the paper.

The next core component of [MZ22] is an algorithm $\mathcal{A}_1(y,z)$ that runs $\mathcal{A}_0(y,z)$ for a number T (to be determined later) of times to get a list L of outputs (some may be repeated multiple times), which provides an empirical distribution $\tilde{\mathcal{D}}$ of the distribution $\mathcal{D}$. When $T \gg 1/\epsilon$, the correct answer $(ab) \star x$ will be on the list with an overwhelming probability. Then for each $w \in L$, they run $\mathcal{A}_0(x,w)$ for T times. When $w = (ab) \star x$ then the resulting distribution will be the same as $\mathcal{D}$. The idea is that if $w \neq (ab) \star x$ then we would like to eliminate it from the list, but this does not always work. We will use a similar "shifting" approach here, but our algorithm is somewhat different and our analysis is considerably different from [MZ22].

One of the main insights of [MZ22] is that the only obstruction is due to small subgroups. We briefly explain this now.

Define $\mathcal{D}_w$ to be the output distribution of $\mathcal{A}_0(x, w)$, and let $\tilde{\mathcal{D}}_w$ be an empirical distribution of $\mathcal{D}_w$ obtained by taking T samples from $\mathcal{D}_w$. Let $\Delta(\cdot, \cdot)$ be the statistical distance function. [MZ22] considers the distance function $\|\mathcal{D} - \mathcal{D}'\|_\infty = \max_{u \in \mathcal{X}} |\mathsf{Pr}[u \leftarrow \mathcal{D}] - \mathsf{Pr}[u \leftarrow \mathcal{D}']|$. Let $L'' \subseteq G$ be the set of $g \in G$ such that $\Delta(\mathcal{D}_{g \star x}, \mathcal{D}) \leq \delta/2$. Let H be the subgroup of G generated by L''. Montgomery and Zhandry show (Lemma 15 of [MZ22]) that if $\delta \leq \epsilon^4/8$ then $|H| < 1/\epsilon + 1$.

The full specification of algorithm $\mathcal{A}_1(y, z)$ is as follows:

1. Run $\mathcal{A}_0(y, z)$ for T times to get a list L of outputs and an estimate $\tilde{\mathcal{D}}$ of the distribution $\mathcal{D}$.
2. Set $L' = \{\}$.
3. For each $w \in L$, run $\mathcal{A}_0(x, w)$ for T times and calculate estimate $\tilde{\mathcal{D}_w}$ of distribution $\mathcal{D}_w$.
4. If $\Delta(\tilde{\mathcal{D}_w}, \tilde{\mathcal{D}}) \leq \delta/2$ then add w to L'.
5. Return L'.

Montgomery and Zhandry then define an algorithm $\mathcal{A}_2(y, z)$ that "fills out" the subgroup so that it outputs the set $\{(gab) \star x : g \in H\}$, where H is the subgroup above mentioned. This algorithm now has overwhelming success. The analysis of $\mathcal{A}_2$ is intricate, and we have intentionally omitted it from our own work. We refer an interested reader to [MZ22].

The full [MZ22] reduction on input a GA-DLog instance $(x, a \star x)$ and with quantum access to circuit $\mathcal{A}$ is as follows:

1. Choose the parameters δ, T.
2. Determine H.
3. Run the algorithm of [GPSV21] with respect to action of G/H on $(G/H) \star x$, using the $\mathcal{A}_2(\cdot, \cdot)$ as the parallelization circuit. Here G/H and $(G/H) \star x$ are represented in $O(1/\epsilon + 1)$ space as cosets/orbits. The algorithm returns the coset aH with noticeable probability.
4. Perform a brute-force search over all elements g within the coset aH, where $a \in aH$, to deduce the group element a using the known set element of $a \star x$.

Our overall approach in this work is similar, although our parameterizations and algorithms are quite different.

Finally, we note that in their published work, Montgomery and Zhandry claim the number of queries to $\mathcal{A}$ is $O\left(1/\epsilon^{13}\right)$. However, there is a miscalculation in the complexity of the algorithm $\mathcal{A}_1$ as presented in [MZ22]. According to their analysis, the algorithm $\mathcal{A}_1$ requires $T^2 + T$ queries of the circuit $\mathcal{A}$, where T is taken to be $T = \tilde{O}(\epsilon^{-8})$. It is crucial to note that the condition $T = \tilde{O}(\epsilon^{-8})$ is necessary to ensure that the subgroup generated by the error terms of $\mathcal{A}_1$ approximates ϵ^{-1}. However, due to this requirement, it actually implies that $\mathcal{A}_1$ performs $\tilde{O}(\epsilon^{-16})$ queries to $\mathcal{A}$ instead of the originally stated $\tilde{O}(\epsilon^{-8})$. Consequently, when provided with a GA-CDH oracle with a success rate of ϵ, solving a GA-DLog problem using $\mathcal{A}_1$ would actually require $\tilde{O}(\epsilon^{-21})$ queries to the CDH oracle, as opposed to the claim of $\tilde{O}(\epsilon^{-13})$ made in the paper.

1.3 Technical Overview

We show a new approach to the problem based on thresholding. For a GA-CDH challenge $(a \star x, b \star x)$, we essentially show that there is a set of "heavy" elements that contains the required value $(ab) \star x$ that can be accurately computed using a sufficient number of queries to the oracle. Unlike in [MZ22], we can show that, across different queries the same set of elements, up to shifting by some value, always shows up in an output set. This makes our statistical analysis much easier and more lightweight– [MZ22] has to use a number of complicated theorems from algebra, while the most complicated math we use is a simple Chernoff bound–as well as dramatically more efficient. We outline the steps in our reduction in the remainder of this subsection.

We also assume in this overview we are working with a regular group action $(G, \mathcal{X}, \star)$ with origin element x. Consider a GA-CDH oracle $\mathcal{A}$ that outputs the correct set element with probability ϵ. We show how to use $\mathcal{A}$ to build an algorithm that outputs either the correct set element or all elements in a coset of a subgroup containing the correct set element with extremely high probability; from there, we can apply the work of [GPSV21] and [MZ22] to complete the GA-CDH to GA-DLog reduction.

The simple randomized self-reduction. As we outlined earlier, one of the core algorithms in [MZ22] and in our work is the simple random self-reduction $\mathcal{A}_0$ for GA-CDH instances on a group action. Suppose we are given GA-CDH challenge set elements $(y = a \star x, z = b \star x)$ and want to query $\mathcal{A}$ to output $(ab) \star x$. $\mathcal{A}$ could just refuse to work on certain inputs; its success probability is over all combinations of set elements. However, we can have an efficient self-reduction: by randomly selecting g and h from the group G and calculating $(gh)^{-1} \star \mathcal{A}(g \star x, h \star y)$, we obtain the correct result if and only if $\mathcal{A}$ correctly evaluates the query. Furthermore, since $g \star x$ and $h \star y$ constitute uniformly random and independent set elements (we assume here that the group action is regular), we obtain the distribution that represents the "average" output of the adversary.

Following [MZ22] we refer to this algorithm as $\mathcal{A}_0$. The distribution resulting from $\mathcal{A}_0(x, x)$ is denoted as $\mathcal{D}$, and can be viewed as the "reference" distribution. We adopt straightforward proofs from [MZ22] to establish that, for any $g \in G$, $g \star \mathcal{D} = A_0(x, g \star x)$, which we denote as $\mathcal{D}_g$.

In essence, we are asserting that if we modify the input to $\mathcal{A}_0$, the distribution of the adversary's output will shift accordingly. This is because the inputs to $\mathcal{A}$ from $\mathcal{A}_0$ are entirely randomized, preventing $\mathcal{A}$ from engaging in any strategic maneuvers or attempts to deceive.

Approximating the oracle's success probability ϵ. In contrast to [MZ22], our reduction commences with a concrete approximation of the given GA-CDH oracle's success probability, denoted as ϵ. Establishing a tight lower bound on ϵ, denoted by ϵ_{min}, is a crucial step in our reduction process. This lower bound serves a vital role in enabling a *fully black-box reduction*, a distinction from [MZ22].

To determine ϵ_{min}, we execute $\mathcal{A}_0(x, x)$ a sufficient number of times and count the instances where x is the output. Given that we are aware of the value of x and its correctness as a solution to a CDH query on (x, x), this procedure is straightforward and efficient. For the sake of simplicity in this overview, we assume that $\epsilon = \epsilon_{\mathsf{min}}$ and that we know the value of ϵ. However, it's important to note that our results do not hinge on this assumption being the case.

Building a threshhold list. As per our assumption above, we know the adversary $\mathcal{A}$ succeeds with probability ϵ, and we can leverage $\mathcal{A}_0$ to ensure this success rate on any query. Our next goal is to show we can, requiring roughly (asymptotically) $1/\epsilon^3$ queries to $\mathcal{A}_0$, "threshhold" the output such that, in response to any CDH challenge query (y, z), we generate a list of precisely I elements. Here, I is a fixed integer where $I = O\left(\frac{1}{\epsilon}\right)$, and this list consists of the top I elements from the distribution $\mathcal{A}_0(y, z)$.[1]

By assumption, we have that the adversary must output the correct answer to the CDH challenge with probability ϵ. Suppose we rank the elements output by $\mathcal{D}$–the output distribution of $\mathcal{A}_0$ by likelihood of appearing. The most likely element x_1 occurs with probability p_1, the second most likely element x_2 occurs with probability p_2, and so forth. If x_c is the correct element, we claim that there must be some elements x_i, x_{i+1} where $i \geq c$ and $p_i - p_j \geq k\epsilon^2$ for some constant k and some $i \leq \frac{2}{\epsilon}$. This follows from summing the p_is using the Gaussian summation formula (see, we told you, simple math!): if there is no gap of the appropriate size, the probabilities will sum to something larger than 1.

It turns out that if we sample $\mathcal{A}_0(x, x)$ enough times, we can find I and this gap by just seeing where a large gap lies. We use Chernoff bounds to show that asymptotically $1/\epsilon^3$ samples are enough to do this, and this turns out to be the bulk of the writing in our proof. If we provided the Chernoff bounds out of thin air, then our already short proof would be extremely short. We call this algorithm for gap-finding $\mathcal{A}_I$, and we note that it works, for any input values (x, y). We do, however, write down and keep track of I for our future algorithms, because it could be possible that there are two gaps of similar size, and we want to make sure that we use the same set of elements (of the same size) every time we attempt to threshold.

A "shifting" algorithm. At this point, we borrow conceptually from [MZ22], but our algorithms will be different. We define a new algorithm which we refer to as $\mathcal{A}_1$, which is conceptually similar to the algorithm of the same name from [MZ22]. $\mathcal{A}_1$ does almost the exact same thing as $\mathcal{A}_I$, except it uses the knowledge of I to always output I elements. So, $\mathcal{A}_1(y, z)$ outputs a set of I elements $z_1, ..., z_I$, one of which must be the correct GA-CDH answer. Suppose, for each z_i, we compute $\mathcal{A}_1(x, z_i)$. For the correct z_i, we know that $\mathcal{D}_{y,z} = \mathcal{D}_{x,z_i}$, where we are overloading the notation of $\mathcal{D}$ in the natural way, because $\mathcal{A}_0$ is "shift invariant." Hence, it is very straightforward to see that, with an overwhelming chance, we have $\mathcal{A}_1(y, z) = A_1(x, z_i)$, if z_i is the correct answer. As a result, after the

[1] Note this is very different from the algorithm of the same name in [MZ22].

execution of $\mathcal{A}_1(x, z_i)$ for each i, we can eliminate all z_i for which $\mathcal{A}_1(y, z) \neq A_1(x, z_i)$ from our candidate list of correct solutions. We will call this algorithm $\mathcal{A}_2$ and denote the resulting list L.

The authors of [MZ22] opt for a more intricate shifting and pruning algorithm. Without thresholding, they cannot ensure that each "run" of their $\mathcal{A}_1$ will consistently produce the same list of elements (although they may be shifted). This crucial distinction is the primary reason why our algorithm stands out as simpler and significantly more efficient than theirs.

Why we have a full subgroup. The primary challenge, which is also a key source of inefficiency in [MZ22], revolves around the necessity of finding a "complete" coset (i.e. the set elements generated by $H \star x$ for some subgroup H). In the reduction, this step is indispensable, as it paves the way for the application of Shor's algorithm in the final step. However, in our case here, it is straightforward to show that the list L constitutes a complete coset already. To see this, every element in L needs to be "shift invariant" onto the set L with respect to $\mathcal{A}_1$: in other words, we have $\mathcal{A}_1(x, L) = L$, or else $\mathcal{A}_2$ would have pruned these elements. It is straightforward to derive a contradiction if L is not a complete coset: we either break the "shift invariance" of $\mathcal{A}_0$ and $\mathcal{A}_1$, or the fact that $\mathcal{A}_2$ should have eliminated certain elements.

Cleaning up. Now that we have outlined how our improved reduction outputs a set L that is a complete coset containing the correct solution to a GA-CDH instance, all that remains is to show that we can clean up correctly. We do this exactly as in [MZ22] and [GPSV21]. We can use L and Shor's algorithm to find a subgroup H that generates L from the correct solution, run the core algorithm from [GPSV21] on the induced group action $(G/H, G/H \star x, \star)$, and then "brute force" over all elements of H to get a final answer.

Our total running time is proportional to $1/\epsilon^4$ and some polynomial in $\log(|G|)$, which is a substantial improvement over $\tilde{O}(\epsilon^{-21})$ from the previous work.

2 Preliminaries

We begin by defining basic background material. A reader knowledgeable in group actions and cryptography may safely skip this section.

2.1 Cryptographic Group Actions

We define cryptographic group actions following Alamati *et al.* [ADMP20], which are based on those of Brassard and Yung [BY91] and Couveignes [Cou06]. Our presentation here is based on that of [MZ22].

Definition 1. (Group Action) *A group G is said to* act on *a set $\mathcal{X}$ if there is a map $\star : G \times \mathcal{X} \to \mathcal{X}$ that satisfies the following two properties:*

1. *Identity: If e is the identity of G, then $\forall x \in \mathcal{X}$, we have $e \star x = x$.*
2. *Compatibility: For any $g, h \in G$ and any $x \in \mathcal{X}$, we have $(gh) \star x = g \star (h \star x)$.*

We may use the abbreviated notation $(G, \mathcal{X}, \star)$ to denote a group action. We extensively consider group actions that are *regular*:

Definition 2. *A group action $(G, \mathcal{X}, \star)$ is said to be* regular *if, for every $x_1, x_2 \in \mathcal{X}$, there exists a* unique *$g \in G$ such that $x_2 = g \star x_1$.*

We emphasize that most results in group action-based cryptography have focused on regular actions. As emphasized by [ADMP20], if a group action is regular, then for any $x \in \mathcal{X}$, the map $f_x : g \mapsto g \star x$ defines a bijection between G and $\mathcal{X}$; in particular, if G (or $\mathcal{X}$) is finite, then we must have $|G| = |\mathcal{X}|$.

In this paper, unless we specify otherwise, we will work with *effective* group actions (EGAs). An effective group action $(G, \mathcal{X}, \star)$ is, informally speaking, a group action where all of the (well-defined) group operations and group action operations are efficiently computable, there are efficient ways to sample random group elements, and set elements have unique representation. Since the focus of this paper is on abelian group actions in a quantum world, we note that we can efficiently map any abelian group to Z_p for some integer p , and all of the less obvious properties needed for EGAs follow automatically. Formally speaking, we define an *effective group action* (EGA) as follows:

Definition 3. (Effective Group Action) *A group action $(G, \mathcal{X}, \star)$ is* effective *if the following properties are satisfied:*

1. *The group G is finite and there exist efficient (PPT) algorithms for:*
   (a) *Membership testing, i.e., to decide if a given bit string represents a valid group element in G.*
   (b) *Equality testing, i.e., to decide if two bit strings represent the same group element in G.*
   (c) *Sampling, i.e., to sample an element g from a distribution $\mathcal{D}_G$ on G. In this paper, We consider distributions that are statistically close to uniform.*
   (d) *Operation, i.e., to compute gh for any $g, h \in G$.*
   (e) *Inversion, i.e., to compute g^{-1} for any $g \in G$.*
2. *The set $\mathcal{X}$ is finite and there exist efficient algorithms for:*
   (a) *Membership testing, i.e., to decide if a bit string represents a valid set element.*
   (b) *Unique representation, i.e., given any arbitrary set element $x \in \mathcal{X}$, compute a string $\hat{x}$ that canonically represents x.*
3. *There exists a distinguished element $x_0 \in \mathcal{X}$, called the* origin, *such that its bit-string representation is known.*
4. *There exists an efficient algorithm that given (some bit-string representations of) any $g \in G$ and any $x \in \mathcal{X}$, outputs $g \star x$.*

2.2 Computational Problems

We next define problems related to group action security. We emphasize that we are defining *problems* here and not *assumptions* because these are easier to use in reductions. Again our presentation is based on that of [MZ22].

Definition 4. (Group Action Discrete Logarithm (DLog)) *Given a group action* $(G, \mathcal{X}, \star)$ *and distributions* $(\mathcal{D}_{\mathcal{X}}, \mathcal{D}_G)$*, the group action discrete logarithm problem is defined as follows: sample* $g \leftarrow \mathcal{D}_G$ *and* $x \leftarrow \mathcal{D}_{\mathcal{X}}$*, compute* $y = g \star x$*, and create the tuple* $T = (x, y)$*. We say that an adversary solves the group action discrete log problem if, given* T *and a description of the group action and sampling algorithms, the adversary outputs* g*.*

Definition 5. (Group Action Computational Diffie-Hellman (CDH)) *Given a group action* $(G, \mathcal{X}, \star)$ *and distributions* $(\mathcal{D}_{\mathcal{X}}, \mathcal{D}_G)$*, the group action CDH problem is defined as follows: sample* $g \leftarrow \mathcal{D}_G$ *and* $x, x' \leftarrow \mathcal{D}_{\mathcal{X}}$*, compute* $y = g \star x$*, and create the tuple* $T = (x, y, x')$*. We say that an adversary solves the group action CDH problem if, given* T *and a description of the group action and sampling algorithms, the adversary outputs* $y' = g \star x'$*.*

Remark 1. The above definitions allow for different distributions $\mathcal{D}_{\mathcal{X}}$ on $\mathcal{X}$. In particular, $\mathcal{D}_{\mathcal{X}}$ could be uniform over $\mathcal{X}$, or it could be a singleton distribution that places all its weight on a single fixed x. Whether x is fixed or uniform potentially changes the the nature of these problems (see [BMZ19] for an exploration in the group-based setting). Looking ahead, as in [MZ22], our reduction between DLog and CDH will preserve x, and therefore it works no matter how x is modeled.

2.3 Chernoff Bounds

In our forthcoming argument, we will rely on Chernoff bounds. To this end, we present a specific formulation of a Chernoff bound below.

Theorem 1. *Let* $X = \sum_{i=1}^{T} X_i$*, where* X_i *are independent random variables with a Bernoulli distribution with* $\Pr[X_i = 1] = p_i$ *and* $\Pr[X_i = 0] = 1 - p_i$*. Let* $\mu = \mathbb{E}[X] = \sum_{i=1}^{T} p_i$*. Then, we have*

$$\Pr\left[X - \mu \geq \eta\mu\right] \leq e^{-\mu\eta^2/(2+\eta)}$$

for any $\eta \geq 0$*, and*

$$\Pr\left[X - \mu \leq -\eta\mu\right] \leq e^{-\mu\eta^2/2}$$

for any $\eta \in (0, 1)$*.*

If $p_i = p$ for all $i \in [T]$ for some $p \in [0, 1]$, then we can restate the inequalities as follows:

$$\Pr\left[X - Tp \geq \eta Tp\right] \leq e^{-Tp\eta^2/(2+\eta)}$$

for any $\eta \geq 0$, and

$$\Pr\left[X - Tp \leq -\eta Tp\right] \leq e^{-Tp\eta^2/2}$$

for any $\eta \in (0,1)$. Moreover, for any $\eta \in (0,1)$ we have

$$\Pr\left[|X - Tp| \geq \eta Tp\right] \leq 2e^{-Tp\eta^2/3}.$$

3 The Main Reduction

We state our main result.

Theorem 2. *Let $(G, \mathcal{X}, \star)$ be an effective group action. If DLog is post-quantum hard in $(G, \mathcal{X}, \star)$, then so is CDH. More precisely, given a CDH adversary $\mathcal{A}$ there exists an oracle algorithm $\mathcal{R}^{\mathcal{A},(G,\mathcal{X},\star)}(y)$ that runs in time $O\left(\mathsf{poly}(\log|G|)/\epsilon^4\right)$ with $\mathsf{poly}(\log|G|)/\epsilon^4$ queries to $\mathcal{A}$ and the group action $(G, \mathcal{X}, \star)$ such that*

$$\mathsf{Adv}^{(G,\mathcal{X},\star)}_{\mathsf{DLog}}\left(\mathcal{R}^{\mathcal{A},(G,\mathcal{X},\star)}\right) \geq 0.99,$$

where $\epsilon := \mathsf{Adv}^{(G,\mathcal{X},\star)}_{\mathsf{CDH}}(\mathcal{A})$.

The running time and number of calls to $\mathcal{A}$ of the black-box reduction $\mathcal{R}$ depend on the success probability ϵ of $\mathcal{A}$. Nontheless, we are not required to know ϵ in advance and the estimation is also a part of our reduction. The remainder of this section is devoted to proving Theorem 2.

3.1 Preparation

Our basic setup very closely mirrors that of [MZ22], so we borrow their presentation for much of the beginning of this section. Let $x \in \mathcal{X}$ be a fixed set element. Define CDH to be the function which correctly solves CDH relative to x: $\mathsf{CDH}(a \star x, b \star x) = (ab) \star x$. We extend the oracle CDH to accept a vector of elements as input, operating as follows: $\mathsf{CDH}(a_1 \star x, \cdots, a_n \star x) = (a_1 \cdots a_n) \star x$. Moreover, we permit CDH to process distribution(s) over the set $\mathcal{X}$ as input. In such cases, CDH will naturally yield a corresponding distribution as its output. We will later use a very similar argument as in [MZ22] in Sect. 3.9 and explain how to extend our reduction to non-regular abelian actions.

Let $a, b \in G$ be group elements, and let $y = a \star x$ and $z = b \star x$. Suppose $\mathcal{A}$ is an efficient (quantum) algorithm such that

$$\epsilon := \mathsf{Adv}^{(G,\mathcal{X},\star)}_{\mathsf{CDH}}(\mathcal{A}) = \Pr_{a,b \in G}[\mathcal{A}(x, a \star x, b \star x) = (ab) \star x]$$

is a non-negligible function in the security parameter, where a and b are random elements in G, and the probability is over the randomness of a and b and $\mathcal{A}$.

Our goal is to turn $\mathcal{A}$ into a quantum algorithm for discrete logarithms. As a first step, we will introduce the basic random self-reduction for CDH from [MZ22].

Algorithm $\mathcal{A}_0$.

- On input $y = a \star x, z = b \star x$, choose elements $a', b' \in G$ uniformly at random.
- Assign $(y', z') \leftarrow (a' \star y, b' \star z)$.
- Run $w' \leftarrow \mathcal{A}(x, y', z')$.
- Output $w \leftarrow (a'b')^{-1} \star w'$.

Note that each run of $\mathcal{A}_0$ runs $\mathcal{A}$ exactly once, and uses a constant number of group action operations. This reduction preserves the correctness of $\mathcal{A}$, since, if $\mathcal{A}$ is correct, then we output

$$w = (a'b')^{-1} \star \mathsf{CDH}\left((a'a) \star x, (b'b) \star x\right) = (a'b')^{-1} (aa'bb') \star x = (ab) \star x$$

which is the correct output for CDH on input (y, z). Furthermore, as the set elements y' and z' are uniformly distributed over $\mathcal{X}$, the success rate of A_0 will be independent of the input.

Let $\mathcal{D}$ represent the output distribution of $\mathcal{A}_0(x, x)$. While the answer to $x = \mathsf{CDH}(x, x)$ is trivial, the distribution $\mathcal{D}$ provides crucial clues for our analysis.

Lemma 1. *(Lemma 10, [MZ22])* $\Pr[x \leftarrow \mathcal{D}] = \epsilon$.

Proof. Recall that $\mathcal{D}$ is the distribution $\mathcal{A}_0(x, x)$. A_0 on input (x, x) calls $\mathcal{A}(a' \star x, b' \star x)$ for random $a', b' \in G$. With probability ϵ, $\mathcal{A}(a' \star x, b' \star x)$ returns $(a'b') \star x$, and in this case we have $w = x$ as desired. □

We next generalize our notation. For any $y, z \in X$ where $y = a \star x$ and $z = b \star x$ for some $a, b \in G$, let $\mathcal{D}_{y,z}$ be the distribution of outputs of $\mathcal{A}_0(y, z)$.

Lemma 2. *(Lemma 11, [MZ22]) For every $y, z \in \mathcal{X}$ such that there exist $a, b \in G$ where $y = a \star x$ and $z = b \star x$, $\mathcal{D}_{y,z} = \mathsf{CDH}(y, z, \mathcal{D})$, where $\mathsf{CDH}(\cdot, \cdot, \cdot)$ is the 3-way CDH function. In other words, $\mathcal{A}_0(a \star x, b \star x)$ is identically distributed to $(ab) \star \mathcal{A}_0(x, x)$.*

Proof. Fix $a, b \in G$. Consider the probability that $\mathcal{A}_0(a \star x, b \star x)$ outputs w:

$$\begin{aligned}
\Pr[\mathcal{A}_0(a \star x, b \star x) = w] &= \Pr_{a', b' \in G}[(a'b')^{-1} \star \mathcal{A}((aa') \star x, (bb') \star x) = w] \\
&= \Pr_{a', b' \in G}[\mathcal{A}((aa') \star x, (bb') \star x) = (a'b') \star w] \\
&= \Pr_{a'', b'' \in G}[\mathcal{A}(a'' \star x, b'' \star x) = (a''b'' (ab)^{-1}) \star w] \\
&= \Pr[\mathcal{A}_0(x, x) = (ab)^{-1} \star w]
\end{aligned}$$

Thus, $\mathcal{A}_0(a \star x, b \star x)$ is just the distribution $\mathcal{A}_0(x, x)$, but shifted by ab. □

For some intuition on this lemma, we emphasize that A_0 completely re-randomizes the output that the adversary sees. In other words, on an input $(a \star x, b \star x)$ to A_0, the adversary sees a CDH tuple $(x, (ga) \star x, (hb) \star x)$ for uniformly random group elements g and h. Then, A_0 takes whatever set element

x' that the adversary returns and outputs $(gh)^{-1} \star x'$. Note that, even if the adversary could solve GA-DLog, it couldn't output a constant element: even if it can solve for (ga) and (hb), it information-theoretically doesn't know what a is (or g for that matter). For instance, if the adversary got $(x, c \star x, d \star x)$ and always tried to output $(cd)^{-1} \star z$ for some fixed set element z, A_0 wouldn't actually output a constant element: if $c = g$ and $d = h$, then A_0 would just output z (i.e. the case where $a = b = 1$), but if $c = ga$ and $d = hb$, then A_0 would output $(ab) \star z$, as the lemma states.

Using this "shift invariance" we can define $\mathcal{D}_w := \mathcal{D}_{w,x} = \mathcal{D}_{x,w} = \mathcal{D}_{y,z}$, if $\mathsf{CDH}(y, z) = w$. Lemma 2 shows that $\mathcal{D}_{y,z}$ outputs $\mathsf{CDH}(y, z)$ with probability ϵ. Thus, by running $\mathcal{A}_0$ many times, the right answer is almost certainly amongst the list of outputs. However, to amplify the success probability, we would need to know which element of the list of outputs is the correct answer; we cannot determine this yet.

3.2 Estimating ϵ

At this point, we deviate from the approach taken in [MZ22]. We will need to have a precise estimation of a lower bound for ϵ in our later algorithms; luckily, this is easy enough for us to compute. Although we need to use some statistical tests, our approach is straightforward: we just run $\mathcal{A}_0(x, x)$ "enough" times and keep track of how many times we get x as an output. We generate a (w.h.p.) lower bound for ϵ which we call ϵ_{min}.

Algorithm $\mathcal{A}_\epsilon(\lambda, \lambda')$ (Estimating ϵ).

- On input (security) parameters λ, λ' where λ' can be chosen linearly in the security parameter λ, do the following:
- Set $c = \lambda^2 \lambda'$.
- Set $T = 0, i = 0$.
- While $i < c$:
  - Run $x' \leftarrow \mathcal{A}_0(x, x)$.
  - If $x' == x$ then $i++$.
  - $T++$.
- Output $\epsilon_{\mathsf{min}} = \left(1 - \frac{1}{\lambda}\right) \frac{c}{T}$.

We next prove some bounds on our estimation of ϵ_{min}. We use a simple Chernoff bound.

Lemma 3. *Let all parameters be defined as above. When $\lambda > 2$, except with probability that decays exponentially in λ', we have $\epsilon_{\mathsf{min}} < \epsilon$.*

Proof. We assume that $\epsilon T < c$, or otherwise our bound holds trivially. Note that

$$\Pr[\epsilon_{\mathsf{min}} \geq \epsilon] = \Pr\left[\left(1 - \frac{1}{\lambda}\right) \cdot \frac{c}{T} \geq \epsilon\right] = \Pr\left[c - \epsilon T \geq \frac{c}{\lambda}\right]$$

Using a Chernoff bound, we have

$$\Pr\left[c - \epsilon T \geq \eta \epsilon T\right] \leq e^{-\epsilon T \eta^2/2+\eta}.$$

for any $\eta > 0$. Suppose we set $\eta \epsilon T = \frac{c}{\lambda}$ for our parameter λ. If we continue to assume that $\epsilon T < c$, we have $\eta \geq \frac{1}{\lambda}$. This gives us

$$\Pr\left[c - \epsilon T \geq \eta \epsilon T\right] \leq e^{-\epsilon T \eta^2/2+\eta} \leq e^{-\frac{c}{\lambda} \cdot \frac{\eta}{\eta+2}} \leq e^{-\frac{c}{3\lambda^2}}.$$

Setting $c = \lambda^2 \lambda'$ makes this equation decay exponentially in λ', as desired. □

We can think of λ and λ' as essentially security parameters. We leave these undefined for now because we will need to make the error probability in our final algorithm dependent on the group size. We now prove an upper bound on ϵ.

Lemma 4. *Let all parameters be defined as above, and let $\lambda \geq 5$. Except with probability that decays exponentially in λ', we have $\epsilon < \left(1 + \frac{3}{\lambda}\right) \epsilon_{\mathsf{min}}$.*

Proof. When $\epsilon T \leq c$, the statement holds trivially if $\lambda \geq 2$. This is because $(1 + \frac{3}{\lambda})(1 - \frac{1}{\lambda}) > 1$ when $\lambda \geq 2$. Hence, we assume $\epsilon T > c$.

At this point, the result follows from another Chernoff bound. Note that

$$\Pr\left[\epsilon T - c \geq \eta \epsilon T\right] \leq e^{-\epsilon T \eta^2/2}.$$

for any $\eta \in (0, 1)$. Recall that $\epsilon_{\min} = \left(1 - \frac{1}{\lambda}\right) \frac{c}{T}$, and therefore we have that $c = \frac{\epsilon_{\min} T}{1 - \frac{1}{\lambda}}$. If we take $\eta = \frac{2\lambda - 3}{\lambda^2 + 2\lambda - 3}$, then we have

$$\Pr\left[\epsilon T - c \geq \eta \epsilon T\right] = \Pr\left[\epsilon T - \frac{\epsilon_{\min} T}{1 - \frac{1}{\lambda}} \geq \frac{2\lambda - 3}{\lambda^2 + 2\lambda - 3} \epsilon T\right].$$

Some basic algebra gives us that

$$\Pr\left[\epsilon T - \frac{\epsilon_{\min} T}{1 - \frac{1}{\lambda}} \geq \frac{2\lambda - 3}{\lambda^2 + 2\lambda - 3} \epsilon T\right] = \Pr\left[\epsilon \geq \left(1 + \frac{3}{\lambda}\right) \epsilon_{\mathsf{min}}\right]$$

as desired. Thus, by taking $\eta = \frac{2\lambda - 3}{\lambda^2 + 2\lambda - 3}$, with the Chernoff bound, we have

$$\Pr\left[\epsilon \geq \left(1 + \frac{3}{\lambda}\right) \epsilon_{\mathsf{min}}\right] \leq e^{-\frac{T \epsilon \eta^2}{2}} \leq e^{-\frac{T \epsilon}{2\lambda^2}} \leq e^{-\frac{c}{2\lambda^2}},$$

where $\eta \in (0, 1)$ and $\eta > 1/\lambda$ when $\lambda \geq 5$. Since $c = \lambda^2 \lambda'$, the statement holds except with probability $e^{-\frac{\lambda'}{2}}$. □

We can now easily determine the running time of $\mathcal{A}_\epsilon$.

Lemma 5. *Let all parameters be defined above, and let $\lambda \geq 5$. Algorithm $\mathcal{A}_\epsilon$ terminates in time $O\left(\frac{1}{\epsilon} \lambda^2 \lambda'\right)$ with probability one minus a function exponentially decaying in λ'.*

Proof. This follows as an immediate corollary of Lemma 4. □

We now know that we can closely estimate ϵ, and we can find such an estimate ϵ_{min} efficiently.

3.3 Thresholding

We know from earlier that $A_0(x, x)$ outputs elements according to some "true" distribution $\mathcal{D}$, and that using different set elements instead of x only shifts this distribution. We know that, by assumption $A_0(x, x)$ outputs x with probability $\epsilon > \epsilon_{\mathsf{min}}$, which is a fact we will use extensively. Below, we formally define some properties of this distribution that will be useful to us for building an algorithm.

Let the distribution $\mathcal{D}$ be supported on $x_1, x_2, x_3, \ldots$ in $\mathcal{X}$ such that, writing $p_i = \mathsf{Pr}(x_i \leftarrow \mathcal{D})$ we have $p_1 \geq p_2 \geq p_3 \geq \cdots$. Then $p_1 \geq \epsilon$. The following result shows that fairly quickly there is a noticeable "gap" $p_i - p_{i+1}$ that we can use for thresholding. Since we don't exactly know ϵ (and thus, can't use it), we will write the lemmas below for ϵ_{min}, which we know is relatively close to ϵ.

Lemma 6. *Let $p_i = \mathsf{Pr}(x_i \leftarrow \mathcal{D})$ be defined as above, so that $p_1 \geq \epsilon$ and $p_1 \geq p_2 \geq p_3 \geq \cdots$. Let $0 < \epsilon_{\mathsf{min}} \leq \epsilon$ be any real number. Let i_0 be the smallest integer such that $p_{i_0} > \epsilon_{\min}$ and $p_{i_0+1} \leq \epsilon_{\min}$. Let $\ell > 0$ be an integer such that ℓ is divisible by 2, and let $\delta \in (0, 1)$ be a real number. If $\ell\left(\epsilon_{\mathsf{min}} - \frac{\ell\delta}{2}\right) > 1$, then there is some integer $i < i_0 + \ell$ such that $p_i \leq \epsilon$ and $p_i - p_{i+1} \geq \delta$.*

Proof. Because the p_i are probabilies, we know that $\sum_{i=1}^{|G|} p_i = 1$. Let i_0 be the smallest integer such that $p_{i_0} > \epsilon_{\mathsf{min}}$ and $p_{i_0+1} \leq \epsilon_{\mathsf{min}}$. Hence $p_{i_0+1} \leq \epsilon$.

If $p_{i_0+1} < \epsilon_{\mathsf{min}} - \delta$ then, since there is some i such that $p_i = \epsilon$, it follows that $p_{i_0} \leq \epsilon$. The result holds in this case by taking $i = i_0$. Hence it suffices to consider the case $p_{i_0+1} \geq \epsilon_{\mathsf{min}} - \delta$. Suppose, for the purposes of contradiction, that for all i such that $i_0 < i < i_0 + \ell$ we have $p_i - p_{i+1} < \delta$.

We have

$$\sum_{i=1}^{i_0+\ell} p_i \leq 1.$$

Since we know that $p_{i_0} > \epsilon_{\mathsf{min}}$, as well as for all $i_0 < i < i_0 + \ell$, $p_i - p_{i+1} < \delta$, we have

$$\sum_{i=1}^{i_0+\ell} p_i > \sum_{i=1}^{i_0} \epsilon_{\min} + \sum_{k=1}^{\ell} p_{i_0+k} > i_0\epsilon_{\min} + \sum_{k=1}^{\ell} (\epsilon_{\min} - k\delta).$$

By the Gauss summation formula, this implies that

$$\sum_{i=1}^{i_0+\ell} p_i > i_0\epsilon_{\mathsf{min}} + \ell\left(\epsilon_{\mathsf{min}} - \frac{(\ell+1)\delta}{2}\right) = (\ell + i_0)\epsilon_{\mathsf{min}} - \ell(\ell+1)\delta/2.$$

However, we have assumed that $\ell\left(\epsilon_{\mathsf{min}} - \frac{\ell\delta}{2}\right) > 1$, which implies that $\sum_{i=1}^{\ell} p_i > 1$. This gives us the desired contradiction and completes the proof. □

Concretely, one can verify that $\delta = \epsilon_{\mathsf{min}}^2/4$ and $\ell = 2\lceil 1/\epsilon_{\mathsf{min}} \rceil$ satisfy the equation $\ell\left(\epsilon_{\mathsf{min}} - \frac{\ell\delta}{2}\right) > 1$. Note also that for $\ell < 2/\epsilon_{\mathsf{min}}$ we have

$$\epsilon_{\mathsf{min}} - \ell\delta > \epsilon_{\mathsf{min}} - \frac{2}{\epsilon_{\mathsf{min}}}\frac{\epsilon_{\mathsf{min}}^2}{4} = \frac{\epsilon_{\mathsf{min}}}{2}.$$

Hence the hard case of thresholding is when there are p_i such that $\epsilon > p_i > \epsilon_{\mathsf{min}}/2$. Since $i_0 \leq 1/\epsilon_{\mathsf{min}}$, we have $i_0 + \ell \leq 3/\epsilon_{\mathsf{min}} + 2$. Since $\epsilon < \epsilon_{\mathsf{min}}(1 + 3/\lambda)$ we have

$$i_0 + \ell \leq 2 + 3/\epsilon_{\mathsf{min}} < 2 + 3(1 + 3/\lambda)/\epsilon = O(1/\epsilon). \quad (1)$$

Remark 2. Note that the above result also applies if p_i are (good) estimations of the true probabilities from some empirical distribution based on a fixed number of samples, up to some (small) margin of error, of course. In practice, the bounds will largely be interchangeable with ϵ rather than ϵ_{min} (up to constant factors, assuming we picked λ and λ' large enough when finding ϵ_{min}). But in our algorithm below we will actually be working with the empirical estimate ϵ_{min} and estimates of the p_i.

3.4 Finding a Gap

Our intuition for how we find a gap is fairly simple: choose some security parameter λ, which will impact the failure probability of our simulation. Then, compute an estimated lower bound ϵ_{min} as we described in the previous subsection and set $\delta = \epsilon_{\mathsf{min}}^2/4$. Then, we will query $\mathcal{A}_0(x, x)$ enough times so that, if there is a gap of size at least $\delta = \frac{\epsilon_{\mathsf{min}}^2}{4}$ between two (estimated) probabilities p_i, p_{i+1}, there will be a noticeable difference in the number of x_i's and x_{i+1}'s that we see over all of the outputs. Then, by standard sampling theorems, if λ'' is large enough, the gap in sampled elements will be at least $k\lambda''$ for some constant k. We can use a similar analysis to show that, with high probability, we don't incorrectly find a small gap either (although we might not find the largest gap). The full algorithm and proof are below.

Algorithm $\mathcal{A}_I(\epsilon_{\mathsf{min}}, \lambda'', \delta, T)$ (Gap-finding algorithm) On input a positive number λ'', which can be chosen linearly to the security parameter λ, and all previous parameters as previously stated. The algorithm $\mathcal{A}_I$ proceeds as follows:

- Initialize an empty database $\mathcal{D}$ consisting of tuples $(x \in \mathcal{X}, t \in \mathbb{Z})$ where $\mathcal{X}$ is the set and t is a nonnegative integer.
- For $(i = 0; i \leq T; i++)$:
  - Set $z_i = \mathcal{A}_0(x, x)$, where each z_i is a "fresh" call of $\mathcal{A}_0(x, x)$.
  - If z_i is the first entry in some tuple $(z_i, t) \in \mathcal{D}$, increment t by 1 in the tuple and update $\mathcal{D}$.
  - If z_i has not yet been added to $\mathcal{D}$, add the tuple $(z_i, 1)$ to $\mathcal{D}$.
- Sort (and relabel) the tuples in $\mathcal{D}$ in *decreasing order of* t, getting a database of tuples (z_1, t_1), (z_2, t_2), ... , such that, for all i, $t_i \geq t_{i+1}$.
- Find the smallest integer i such that $t_{i+1} \leq (\epsilon_{\mathsf{min}} - \delta/2)T$ and $t_i - t_{i+1} \geq T\delta/2$ and output that i. If no such i exists then output $\perp$.

Proving that we find a gap. We next claim that the above algorithm outputs some i with high probability if λ'' is large enough. More precisely, we show that it outputs a gap close to the "best" with high probability, which is good enough for us.

First we need a basic lemma about how well our estimates t_i/T approximate the true values p_i, for values p_i in the worst case zone $\epsilon_{\mathsf{min}} \geq p_i > \epsilon_{\mathsf{min}}/2$ handled by Lemma 6.

Lemma 7. *Let $\epsilon_{\mathsf{min}} \leq \epsilon$ and $\delta = \epsilon_{\mathsf{min}}^2/4$. Let $T = \lambda'' \left(\frac{3072}{\epsilon_{\mathsf{min}}^3}\right)$ for some λ''. Let i be an integer, t_i be the number of times element x is sampled in an experiment where x is sampled T times independently with probability p_i. We have*

1. *If $\epsilon_{\mathsf{min}} \geq p_i$, then $t_i/T - p_i < \delta/8$ holds except for the probability that decays exponentially in λ''.*
2. *Moreover, if $\epsilon_{\mathsf{min}} \geq p_i \geq \epsilon_{\mathsf{min}}/2$, then $|t_i/T - p_i| \leq \delta/8$ with probability that decays exponentially in λ''.*

Proof. Firstly, since

$$\Pr\left[t_i/T - p_i \geq \frac{\delta}{8}\right] = \Pr\left[t_i - p_i T \geq \frac{\delta}{8}T\right] = \Pr\left[t_i - p_i T \geq \left(\frac{\epsilon_{\mathsf{min}}^2}{32 p_i}\right) p_i T\right],$$

by taking $\eta = \frac{\epsilon_{\mathsf{min}}^2}{32 p_i} \geq 0$ for the Chernoff bound, we have

$$\begin{aligned}
\Pr\left[t_i/T - p_i \geq \frac{\delta}{8}\right] &\leq e^{-p_i T \eta^2 \frac{1}{2+\eta}} \\
&= e^{-T \cdot \frac{p_i \cdot \epsilon_{\mathsf{min}}^4}{1024 \cdot p_i^2} \cdot \frac{1}{2+\eta}} \\
&= e^{-\frac{T \cdot \epsilon_{\mathsf{min}}^3}{1024} \cdot \frac{\epsilon_{\mathsf{min}}}{p_i} \cdot \frac{1}{2+\eta}} \\
&= e^{-3\lambda'' \cdot \frac{\epsilon_{\mathsf{min}}}{p_i} \cdot \frac{1}{2+\eta}} \\
&= e^{-3\lambda'' \cdot \frac{\epsilon_{\mathsf{min}}}{p_i} \cdot \frac{32 p_i}{64 p_i + \epsilon_{\mathsf{min}}^2}} \\
&\leq e^{-3\lambda'' \cdot \frac{32 \epsilon_{\mathsf{min}}}{64 \epsilon_{\mathsf{min}} + \epsilon_{\mathsf{min}}^2}} \\
&\leq e^{-3\lambda'' \cdot \frac{32}{65}} \\
&\leq e^{-\lambda''}.
\end{aligned}$$

Similarly, we have

$$\Pr\left[|t_i/T - p_i| \geq \frac{\delta}{8}\right] = \Pr\left[|t_i - p_i T| \geq \frac{\delta}{8}T\right] = \Pr\left[|t_i - p_i T| \geq \left(\frac{\delta}{8 p_i}\right) p_i T\right].$$

Since $\epsilon_{\mathsf{min}} \geq p_i \geq \epsilon_{\mathsf{min}}/2$ we have $1 \leq \epsilon_{\mathsf{min}}/p_i \leq 2$, so

$$\Pr\left[|t_i - p_i T| \geq \left(\frac{\delta}{8 p_i}\right) p_i T\right] \leq \Pr\left[|t_i - p_i T| \geq \left(\frac{\epsilon_{\mathsf{min}}}{32}\right) p_i T\right]$$

By the Chernoff bound, with $\eta = \epsilon_{\mathsf{min}}/32 \in (0,1)$, this is bounded by

$$2e^{-p_i T\eta^2/3} \leq 2e^{-(\epsilon_{\mathsf{min}}/2)T(\epsilon_{\mathsf{min}}/32)^2/3} = 2e^{-\lambda''/2}$$

which proves the result. □

Lemma 8. *Consider all parameters as previously stated. Consider the smallest choice of i such that $p_i - p_{i+1} \geq \delta = \frac{\epsilon_{\mathsf{min}}^2}{4}$ and $p_i \leq \epsilon_{\mathsf{min}}$. It is the case that algorithm $\mathcal{A}_I$ outputs some integer less than or equal to i with probability that decays exponentially in λ''.*

Proof. Let i be the index from Lemma 6, so that $p_i \leq \epsilon$ and $p_i - p_{i+1} \geq \delta$. It follows that $p_i \geq \epsilon_{\mathsf{min}}/2$ and $\epsilon - \delta \geq p_{i+1} \geq \epsilon_{\mathsf{min}}/2 - \delta$. By Lemma 7 (Item 2) we have $|t_i/T - p_i| < \delta/8$.

Since $\epsilon_{\mathsf{min}} \geq p_i > p_{i+1}$, by Lemma 7 (Item 1) we have $t_{i+1}/T - p_{i+1} < \delta/8$. (Recall that p_{i+1} might not less than $\epsilon_{\mathsf{min}}/2$.) It follows that

$$t_i/T - t_{i+1}/T > (p_i - \delta/8) - (p_{i+1} + \delta/8) = (p_i - p_{i+1}) - \delta/4 > \delta/2.$$

This proves the result. □

Lemma 9. *Consider all parameters as previously stated. Let I be the output of algorithm $\mathcal{A}_I$. The probability that $p_I - p_{I+1} \leq \delta/4 = \frac{\epsilon_{\mathsf{min}}^2}{16}$ is a function that decays exponentially in λ''.*

Proof. Suppose $p_I - p_{I+1} \leq \delta/4$ for the purpose of contradiction. Then we claim that $t_I - t_{I+1} \geq T\delta/2$ holds with a negligible chance.

Since $\epsilon_{\mathsf{min}} \geq p_I \geq \epsilon_{\mathsf{min}}/2$, we have $p_{I+1} \geq p_I - \delta/4$ where $\delta = \epsilon_{\mathsf{min}}^2/4$. Since $\epsilon_{\mathsf{min}}/4 \geq \epsilon_{\mathsf{min}}^2/16$ always holds, we have $p_{I+1} \geq \epsilon_{\mathsf{min}}/4$.

Similar to the proof of Lemma 7, write

$$\begin{aligned}\Pr\left[|t_{I+1}/T - p_{I+1}| \geq \frac{\delta}{8}\right] &= \Pr\left[|t_{I+1} - p_{I+1}T| \geq \frac{\delta}{8}T\right] \\ &= \Pr\left[|t_{I+1} - p_{I+1}T| \geq \left(\frac{\delta}{8p_{I+1}}\right)p_{I+1}T\right].\end{aligned}$$

Since $\epsilon_{\mathsf{min}} \geq p_{I+1} \geq \epsilon_{\mathsf{min}}/4$ we have $1 \leq \epsilon_{\mathsf{min}}/p_{I+1} \leq 4$, so

$$\Pr\left[|t_{I+1} - p_{I+1}T| \geq \left(\frac{\delta}{8p_{I+1}}\right)p_{I+1}T\right] \leq \Pr\left[|t_{I+1} - p_{I+1}T| \geq \left(\frac{\epsilon_{\mathsf{min}}}{32}\right)p_{I+1}T\right]$$

By the Chernoff bound, with $\eta = \epsilon_{\mathsf{min}}/32 \in (0,1)$, this is bounded by

$$2e^{-p_{I+1}T\eta^2/3} \leq 2e^{-(\epsilon_{\mathsf{min}}/4)T(\epsilon_{\mathsf{min}}/32)^2/3} = 2e^{-\lambda''/4}.$$

Hence, $|t_{I+1}/T - p_{I+1}| < \frac{\delta}{8}$ with an overwhelming chance. By applying Lemma 7 (Item 2) to the term of index I, we have $|t_I/T - p_I| < \frac{\delta}{8}$ with an overwhelming chance. By combining together, we have $t_I/T - t_{I+1}/T < \delta/4 + p_I - p_{I+1} < \delta/2$ except for a probability that decays exponentially in λ''. That is, $t_I - t_{I+1} \geq T\delta/2$ holds only with a negligible chance, which proves the result. □

To conclude, algorithm $\mathcal{A}_I$ runs in time proportional to $1/\epsilon^3$ and outputs an index $I = O(1/\epsilon)$ such that $p_I \leq \epsilon$ and $p_I - p_{I+1} > \delta/4$.

3.5 Using the Fixed Set of Elements

From the previous section, we know that there will be some index $I \leq \frac{2}{\epsilon_{\mathsf{min}}}$ such that we can efficiently find (in time proportional to $\frac{\lambda}{\epsilon_{\mathsf{min}}^3}$), for some λ'' independent[2] of ϵ_{min}, the set of elements $x_1, \ldots, x_I$ that appear with highest probability. Note that this set is invariant across calls to different inputs to $\mathcal{A}_0$, and we will exploit this in our algorithms.

Algorithm $\mathcal{A}_1(y, z, I, T)$ (Algorithm to find heavy elements)

- Initialize an empty database $\mathcal{D}$ consisting of tuples $(x, t) \in \mathcal{X} \times \mathbb{Z}$ where $\mathcal{X}$ is the set and t is a nonnegative integer.
- For $(i = 0; i \leq T; i++)$:
  - Set $z_i = \mathcal{A}_0(y, z)$, where each z_i is a "fresh" call of $\mathcal{A}_0(y, z)$.
  - If z_i is the first entry in some tuple $(z_i, t) \in \mathcal{D}$, increment t by 1 in the tuple and update $\mathcal{D}$.
  - If z_i has not yet been added to $\mathcal{D}$, add the tuple $(z_i, 1)$ to $\mathcal{D}$.
- Sort (and relabel) the tuples in $\mathcal{D}$ in *decreasing order of* t, getting a database of tuples (z_1, t_1), (z_2, t_2), ... , such that, for all i, $t_i \geq t_{i+1}$.
- Return the set $\{z_1, ..., z_I\}$.

The following lemma shows that if $T = \lambda'' \left(\frac{3072}{\epsilon_{\mathsf{min}}^3}\right)$ then with overwhelming probability Algorithm $\mathcal{A}_1$ does output the I elements that are heaviest, in the sense that the corresponding probabilities p_i are the highest.

Lemma 10. *Consider all parameters as previously stated. Let $T = \lambda'' \left(\frac{3072}{\epsilon_{\mathsf{min}}^3}\right)$ for some λ''. Then algorithm $\mathcal{A}_1$ outputs the I heaviest elements in the distribution except with probability that decays exponentially in λ''.*

Proof. From Lemma 9 we have $p_I - p_{I+1} > \delta/4$. Algorithm $\mathcal{A}_I$ ensures $t_{I+1} \leq (\epsilon_{\mathsf{min}} - \delta/2)T$. Hence, $p_{I+1} < \epsilon_{\mathsf{min}} - \delta/4$. It sufffices to show that the heaviest I elements all appear with frequency strictly larger than $(p_{I+1}+\delta/8)T$ and that the remaining elements all appear with frequency strictly smaller than $(p_{I+1}+\delta/8)T$. Note that $p_{I+1} + \delta/4 < \epsilon_{\mathsf{min}}$.

If $p_i \geq \epsilon \geq \epsilon_{\mathsf{min}} > p_{I+1} + \delta/4$ then

$$\begin{aligned}\Pr[t_i > (p_{I+1} + \delta/8)T] &= \Pr[t_i > (p_i - (p_i - (p_{I+1} + \delta/8)))T] \\ &= \Pr[t_i - p_i T > -(1 - (p_{I+1} + \delta/8)/p_i)p_i T].\end{aligned}$$

The Chernoff bound with $\eta = 1 - (p_{I+1} + \delta/8)/p_i \in (0, 1)$ shows this holds with an overwhelming chance. That is,

$$\Pr[t_i - p_i T \leq -(1 - (p_{I+1} + \delta/8)/p_i)p_i T] \leq e^{-\frac{\eta^2 p_i T}{2}}.$$

The next case is $i \leq I$ where $p_I < \epsilon_{\mathsf{min}}$. We apply the union bound to the $O(1/\epsilon_{\mathsf{min}})$ values of i in this case. Note that $\epsilon > p_i > \epsilon_{\mathsf{min}}/2$ in this case, so we

[2] λ'' may be dependent on $\log |G|$.

can apply Lemma 7. Hence, $|t_i/T - p_i| \leq \delta/8$ with overwhelming probability. $t_i/T \leq p_i + \delta/8$. It follows that for $i \leq I$ we have $t_i/T \geq p_i - \delta/8$, and then $t_i \geq Tp_{i+1} + T\delta/8$ with an overwhelming chance.

Finally, we need to handle the case when $i > I$ (so $p_i \leq p_{I+1}$). For any specific i then the Chernoff bound shows that t_i does not exceed $(p_{I+1} + \delta/8)T$ except with probability that decays exponentially in λ''. But since there are exponentially many such i we need to argue that, for all $j \leq I$ and all $k > I$, the probability that $\mathcal{A}_1$ outputs *any* z_j fewer times than any z_k decays exponentially in λ''.

To handle this, suppose we group each of the z_k's (recall $k > I$) into sets $\tilde{z}_1, \tilde{z}_2, ...$ in the following way: starting with z_{I+1}, add set elements in increasing order to the set $\tilde{z}_1$ as long as the total sum of probabilities of elements in the set is less than p_I. Once $\tilde{z}_1$ is "full", continue this process with the "unused" set elements in increasing order until $\tilde{z}_2$ is "full", and then continue this process until all of the set elements z_k have been placed in a set $\tilde{z}_{k'}$. We note that such a process may not be efficient, but we do not need it to be.

Since the p_i are decreasing, it follows that the probability mass of each set $\tilde{z}_{k'}$ (except perhaps the last one) is at least $p_I/2$. Hence there are a maximum of $\frac{2}{p_I}$ sets $\tilde{z}_{k'}$, or otherwise the sum $\sum_{k=i}^{\infty} p_k > 1$, which is a contradiction. Moreover, note that the probability that some z_k is output more than some z_j is less than the probability that elements in the set $\tilde{z}_{k'}$ containing z_k are output more than the z_j.

Therefore, the probability that $\mathcal{A}_1$ does not output the I heaviest elements is at most $I\frac{2}{p_I}$ multiplied by the probability that $\mathcal{A}_1$ outputs z_{I+1} more than z_I. Since I and p_I are independent of λ'', the statement claimed in the lemma holds. □

Consider the following algorithm, where all parameters are as previously stated. We assume as inputs a CDH challenge $(y, z) = (a \star x, b \star x)$ and all relevant parameters.

Algorithm $\mathcal{A}_2(y, z, I, T)$ *(Pruning)*

- Run the algorithm $\mathcal{A}_1(y, z, I, T)$ from the previous section with oracle calls $\mathcal{A}_0(y, z)$ to get a set of elements $\mathcal{S} = \{z_1, ..., z_I\}$. Record all of these.
- Create a list L of set elements intialized to be empty.
- For each $j \in [1, I]$:
  - Run the algorithm $\mathcal{A}_1(x, z_j, I, T)$ getting a set of elements $\mathcal{S}_j$.
  - If $\mathcal{S}_j = \mathcal{S}$ then add z_j to L.

  Output L.

We next prove a lemma about the running time of this algorithm.

Lemma 11. *Algorithm* $\mathcal{A}_2$ *runs in time* $O\left(\frac{1}{\epsilon_{\min}^4}\right)$ *in* $\epsilon_{\min}$ *and in time polynomial in all other factors.*

Proof. Since $I \leq \frac{c}{\epsilon_{\mathsf{min}}}$ for some constant c, and this algorithm makes I calls to our previous algorithm, it has running time $O\left(\frac{1}{\epsilon_{\mathsf{min}}^4}\right)$, ignorning the λ'' factors, which are technically independent of ϵ_{min} (i.e. λ must be proportional to something in $\log |G|$). □

In the next section we show that L consists of either a single element $(ab) \star x$ or else there is a subgroup H such that $L = \{(hab) \star x : h \in H\}$.

3.6 Proof of Finding the Subgroup

Let $\mathcal{S} = \mathcal{A}_1(x, x)$ be the set of heavy elements output by the gap-finding algorithm on instance x. We have $x \in \mathcal{S}$ with overwhelming probability. For $w \in \mathcal{X}$ let $\mathcal{S}_w = \mathcal{A}_1(x, w)$. Let $L = \mathcal{A}_2(x, x)$ be the list (a subset of $\mathcal{S}$) output by the pruning algorithm. We know that if $x \in L$ then $x \in \mathcal{S}$. For $w \in \mathcal{X}$ let $L_w = \mathcal{A}_2(x, w)$.

For any set $\mathcal{S} = \{x_1, \ldots, x_I\}$ and $g \in G$ define $g \star \mathcal{S} = \{g \star x_1, \ldots, g \star x_I\}$. Ditto for $g \star L$. From lemma 2, we know that $\mathcal{A}_0(x, g \star x) = g \star \mathcal{A}_0(x, x)$.

Lemma 12. *Let notation be as above. The following properties hold:*

1. $\mathcal{S}_{g\star x} = g \star \mathcal{S}$.
2. $\mathcal{A}_1(x, g \star x) = g \star \mathcal{A}_1(x, x)$.
3. $L_{g\star x} = g \star L$.
4. $\mathcal{A}_2(x, g \star x) = g \star A_2(x, x)$.

Proof. To prove the first item, note that $\mathcal{S}_{g\star x}$ is the set of thresholded outputs of $\mathcal{A}_0(x, g \star x)$. But $\mathcal{A}_0(x, g \star x) = g \star \mathcal{A}_0(x, x)$. So $\mathcal{S}_{g\star x} = g \star \mathcal{S}$.

The second part is immediate, since $\mathcal{A}_1(x, g\star x) = \mathcal{S}_{g\star x} = g\star\mathcal{S} = g\star\mathcal{A}_1(x, x)$.

Finally,

$$\begin{aligned} L_{g\star x} &= \{w \in \mathcal{S}_{g\star x} : \mathcal{A}_1(x, w) = \mathcal{A}_1(x, g \star x)\} \\ &= \{w \in \mathcal{S}_{g\star x} : \mathcal{A}_1(x, w) = g \star \mathcal{A}_1(x, x)\} \\ &= \{w \in \mathcal{S}_{g\star x} : g \star \mathcal{A}_1(x, g^{-1} \star w) = g \star \mathcal{A}_1(x, x)\} \\ &= \{w \in g \star \mathcal{S} : \mathcal{A}_1(x, g^{-1} \star w) = \mathcal{S}\}. \end{aligned}$$

Hence $L_{g\star x} = g \star L$. The fourth part, in a similar argument to the second part, is immediate since $\mathcal{A}_2(x, g \star x) = L_{g\star x} = g \star L = g \star \mathcal{A}_2(x, x)$. □

Now we let $H = \{g \in G : g \star x \in L\}$ and we show H is a subgroup. For a set $\mathcal{S}$ we define $H \star \mathcal{S} = \{h \star w : h \in H, w \in \mathcal{S}\}$.

Corollary 1. *Let notation be as above and assume $x \in L$. Let $H = \{g \in G : g \star x \in L\}$. Then H is a subgroup of G and $H \star L = L$. Finally $|H| = |L| \leq |\mathcal{S}| = O(1/\epsilon_{\mathsf{min}})$.*

Proof. Since our group action is regular, we can define $H = \{g \in G : g \star x \in L\}$. Since $x \in L$ we have $1 \in H$ and H is non-empty.

Let $w \in L$ and let $g \in H$ be such that $w = g \star x$. Recall that $L = \mathcal{A}_2(x, x)$ and $\mathcal{A}_2$ takes as input the set $\mathcal{S}$. It then computes all $w \in \mathcal{S}$ such that $\mathcal{A}_1(x, w) = \mathcal{S}$.

By definition of L, we have $L = \mathcal{A}_2(x, x)$. Note that, for any $w \in L$, we have $\mathcal{A}_1(x, w) = \mathcal{A}_1(x, x)$ by definition, and thus we immediately see that $\mathcal{A}_2(x, w) = \mathcal{A}_2(x, x)$, since we will prune identically in both cases. Finally, we have

$$\mathcal{A}_2(x, w) = \mathcal{A}_2(x, g \star x) = g \star \mathcal{A}_2(x, x) = g \star L$$

Hence $g \star L = L$. It follows that $H * L = L$.

(As an aside, going back to $g \star L = L$, by induction we have $g^n \star L = L$ for all integers n. Hence the order of g is at most $|L|$.)

Finally, let $g_1, g_2 \in H$. By definition of $\mathcal{A}_2$, this means $\mathcal{A}_2(x, g_1 \star x) = \mathcal{A}_2(x, g_2 \star x)$. But this implies $g_1 \star L = \mathfrak{g}_2 \star L =: .$ Then $\mathcal{A}_2(x, (g_1 g_2)x) = g_1 \star \mathcal{A}_2(x, g_2 \star x) = g_1 \star L = L$. Hence, $g_1 g_2 \in H$. □

The outcome of all this is that L is a coset of a subgroup H of G. Just like in [MZ22], we can output a complete subgroup H in which our solution is guaranteed to lie.

3.7 Putting It All Together

We are now in a position to state an overall algorithm. We are given a group action $(G, \mathcal{X}, \star)$, a fixed x, and an oracle $\mathcal{A}$. First, we do several precomputations: We run Algorithm $\mathcal{A}_\epsilon$ to compute ϵ_{min}, and then Algorithm $\mathcal{A}_I$ to compute I, and finally we use Algorithm $\mathcal{A}_2$ to compute the list L and hence the subgroup H. The cost of the precomputation is $O\left(\frac{\lambda^2 \lambda'}{\epsilon} + \frac{\lambda''}{\epsilon^4}\right)$, assuming $\lambda > 5$.

When provided with a GA-CDH challenge (x, y, z) we run $\mathcal{A}_2(y, z, I, T)$, which does the pruning to $\mathcal{S}$ and outputs L, which is a coset with respect to a subgroup H.

Lemma 13. *The probability that $\mathcal{A}_2$ does not output a correct L (meaning that L is a complete coset of a subgroup G/H for some subgroup H) decays exponentially in λ, λ', and λ'' assuming $\lambda > 5$.*

Proof. This follows from Lemmas 3, 4 8, 9, 10, and Corollary 1. □

This essentially allows us to minimize our error exponentially by only growing λ' and λ'' linearly.

3.8 Using the Subgroup

At this point, we can go back to the template of [MZ22]. Once we have the appropriate set L, we just need to follow their approach for finishing the overall algorithm. We mirror both their techniques and presentation in this subsection. While we could just cite their results, we present them here for the sake of completeness and point out the text here is only slightly modified from their work.

Removing Superfluous Information. We will next want to run quantum period-finding algorithms which make queries to $\mathcal{A}_2$ on superpositions of inputs. These algorithms, however, assume $\mathcal{A}_2$ is a function. Unfortunately, our algorithm generates significant side information, namely all the intermediate computations used to arrive at the final answer. Fortunately, since our algorithm outputs a single answer with overwhelming probability, we can use the standard trick of purifying the execution of $\mathcal{A}_2$ and then un-computing all the intermediate values. The result is that $\mathcal{A}_2$ is negligibly close to behaving as the function mapping $(y, z) \mapsto H \star \mathsf{CDH}(y, z)$. From now on, we will therefore assume that $\mathcal{A}_2$ is such a function.

Computing H. Given algorithm $\mathcal{A}_2$, we can compute the subgroup H using quantum period-finding [BL95]. Concretely, the function $a \mapsto \mathcal{A}_2(a \star x, x)$ will output $(aH) \star x$, which is periodic with set of periods H. Therefore, applying quantum period finding to the procedure $a \mapsto \mathcal{A}_2(a \star x, x)$ will recover H. This will make $O(\log |G|)$ calls to $\mathcal{A}_2(a \star x, x)$.

Solving DLog in G/H. Notice that $\mathcal{A}_2$ is a (near) perfect CDH-solver, just in the group action corresponding to G/H. Concretely, the group G/H acts on the set $X/H := \{H \star y : y \in X\}$ in the obvious way; the distinguished element of X/H is $H \star x$. Our algorithm $\mathcal{A}_2$ gives a perfect CDH algorithm for this group action: we compute $\mathsf{CDH}(H \star y, H \star z)$ as $\mathcal{A}_2(y', z')$ for an arbitrary $y' \in H \star y, z' \in H \star z$.

We apply Galbraith et al. [GPSV21] to our CDH adversary for $(G/H, X/H)$ to obtain a DLog adversary $\mathcal{B}(gH \star x)$ which computes gH. For completeness, we sketch the idea: Let $\mathbf{a}$ be a set of generators for G/H. Since G is abelian, we can write any g as $\mathbf{a}^{\mathbf{v}}$ for some vector $\mathbf{v} \in \mathsf{Z}_{n_1} \times \cdots \times \mathsf{Z}_{n_k}$ where n_i is the period of a_i. We assume the n_i are fully reduced, so that the choice of $\mathbf{v}$ is unique. Shor's algorithm is used in this step, and we note that Shor's algorithm will not necessarily work if G is not abelian and our group action is not regular, which is why we need this restriction.

The CDH oracle allows, given $h \star (H \star x)$, to compute $h^y \star (H \star x)$ in $O(\log y)$ steps using repeated squaring. Given a DLog instance $g \star (H \star x) = \mathbf{a}^{\mathbf{v}} \star (H \star x)$, we define the function $(\mathbf{x}, y) \mapsto \mathbf{a}^{\mathbf{x}+y\mathbf{v}} \star (H \star x)$, which can be computed using the CDH oracle[3]. Then this function is periodic with period $(\mathbf{v}, -1)$. Running quantum period-finding therefore gives $\mathbf{v}$, which can be used to compute h.

Solving DLog in G We now have an algorithm which solves, with overwhelming probability, DLog in G/H. We now turn this into a full DLog adversary, which works as follows:

- Given $y = c \star x$, first apply the DLog adversary for G/H, which outputs cH.
- For each $a \in cH$ (which is polynomial sized), test if $y = a \star x$. We output the unique such a.

Overall, assuming ϵ is small relative to $\log |G|$, the running time of the algorithm is dominated by the cost of running $\mathcal{A}_2$.

[3] The original paper [GPSV21] needed to solve close vector problems in the relation lattice, but Wesolowski [Wes22] (proof of Theorem 3) has shown that one can bypass this by using the CDH oracle.

3.9 Extending to Non-regular Group Actions

We also borrow the text in this section almost verbatim from [MZ22] for the sake of completeness. The above assumed a regular group action, which captures all the cryptographic abelian group actions currently known. Here, we briefly sketch how to extend to an arbitrary abelian group action. The idea is that, within any ablelian group action, we can pull out a regular group action, and then apply the reduction above.

Concretely, we first consider restricting $(G, X, \star)$ to the orbit of x under G, namely $G \star x$. Let $S \subseteq G$ the the set of a that "stabilizes" x, namely $a \star x = x$. Then S is a subgroup. Moreover, for any $y \in G \star x$, the set of a that stabilize y is also exactly S.

The first step is to compute the (representation of the) subgroup S. Let $f : G \to X$ be defined as $f(a) = a \star x$. Then f is an instance of the abelian hidden subgroup problem with hidden subgroup exactly S. Therefore, we can find S using Shor's quantum algorithm.

Then we can define the new group action $(G/S, G \star x, \star)$, which is a regular abelian group action. CDH in this group action is identical to CDH in the original group action, in that a CDH adversary for one is also a CDH adversary for the other. We can also solve DLog in $(G, X, \star)$ by solving DLog in $(G/S, G \star x, \star)$, and then lifting $a \in G/S$ to $a' = (a, g) \in G$ for an arbitrary $g \in S$.

The main challenge is that our CDH adversary $\mathcal{A}$ may not always output elements in $G \star x$, and it may be infeasible to tell when it outputs an element in $G \star x$ versus a different orbit. Nevertheless, the same reduction as used above applies, and the analysis can be extended straightforwardly but tediously to handle the fact that $\mathcal{A}$ may output elements in different orbits. The rough idea is that $\mathcal{S}$ outputted by $\mathcal{A}_1$ may have pieces from elements from different orbits. But $S \cap G \star x$ is still going to contain a solution, and elements in different cosets will be pruned since they are inherently unreachable. This is enough to ensure that we obtain a near-perfect CDH algorithm on $(G/S)/H$.

Acknowledgements. We thank the anonymous reviewers and Benjamin Wesolowski for comments and suggestions. Galbraith and Lai thank the New Zealand Ministry for Business and Employment for financial support. Yi-Fu Lai is also supported by the European Union (ERC AdG REWORC - 101054911).

References

ADMP20. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 411–439. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_14

AEK+22. Abdalla, M., Eisenhofer, T., Kiltz, E., Kunzweiler, S., Riepel, D.: Password-authenticated key exchange from group actions. In: Lecture Notes in Computer Science, pp. 699–728, Santa Barbara, CA, USA. Springer, Heidelberg, Germany (2022). https://doi.org/10.1007/978-3-031-15979-4_24

BDK+22. Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: generic, simple, and efficient. In: Lecture Notes in Computer Science, pp. 95–126. Springer, Heidelberg, Germany (2022). https://doi.org/10.1007/978-3-031-07085-3_4

BKP20. Beullens, W., Katsumata, S., Pintore, F.: Calamari and Falafl: logarithmic (linkable) ring signatures from Isogenies and Lattices. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 464–492. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_16

BKV19. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 227–247. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_9

BL95. Boneh, D., Lipton, R.J.: Quantum cryptanalysis of hidden linear functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 424–437. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_34

BMM+23. Badrinarayanan, S., Masny, D., Mukherjee, P., Patranabis, S., Raghuraman, S., Sarkar, P.: Round-optimal oblivious transfer and MPC from computational CSIDH. In: Lecture Notes in Computer Science, pp. 376–405. Springer, Heidelberg, Germany (2023). https://doi.org/10.1007/978-3-031-31368-4_14

BMZ19. Bartusek, J., Ma, F., Zhandry, M.: The distinction between fixed and random generators in group-based assumptions. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 801–830. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_27

BY91. Brassard, G., Yung, M.: One-Way group actions. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 94–107. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-38424-3_7

CLM+18. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11274, pp. 395–427. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_15

Cou06. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006). https://eprint.iacr.org/2006/291

DG19. De Feo, L., Galbraith, S.D.: SeaSign: compact Isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 759–789. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_26

DM20. De Feo, L., Meyer, M.: Threshold schemes from Isogeny assumptions. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 187–212. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_7

EKP20. El Kaafarani, A., Katsumata, S., Pintore, F.: Lossy CSI-FiSh: efficient signature scheme with tight reduction to decisional CSIDH-512. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 157–186. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_6

FFK+23. De Feo, L., et al.: SCALLOP: scaling the CSI-FiSh. In: Boldyreva, A., Kolesnikov, V., editors, PKC 2023, vol. 13940 of Lecture Notes in Computer Science, pp. 345–375. Springer (2023). https://doi.org/10.1007/978-3-031-31368-4_13

GPSV21. Galbraith, S., Panny, L., Smith, B., Vercauteren, F.: Quantum equivalence of the DLP and CDHP for group actions. Math. Cryptol. **1**(1), 40–44 (2021)

KLLQ23. Katsumata, S., Lai, Y.F., LeGrow, J.T., Qin, L.: CSI -otter: isogeny-based (partially) blind signatures from the class group action with a twist. In: Lecture Notes in Computer Science, pp. 729–761, Santa Barbara, CA, USA. Springer, Heidelberg, Germany (2023). https://doi.org/10.1007/978-3-031-38548-3_24

LGD21. Lai, Y.-F., Galbraith, S.D., Delpech de Saint Guilhem, C.: Compact, efficient and UC-secure isogeny-based oblivious transfer. In: Lecture Notes in Computer Science, pp. 213–241. Springer, Heidelberg, Germany (2021). https://doi.org/10.1007/978-3-030-77870-5_8

MZ22. Montgomery, H., Zhandry, M.: Full quantum equivalence of group action DLog and CDH, and more. In: Lecture Notes in Computer Science, pp. 3–32. Springer, Heidelberg, Germany (2022). https://doi.org/10.1007/978-3-031-22963-3_1

PR23. Page, A., Robert, D.: Introducing Clapoti(s): evaluating the isogeny class group action in polynomial time. IACR Cryptol. ePrint Arch. 2023/1766 (2023)

Sho94. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, pp. 124–134, Santa Fe, NM, USA, November 20–22 (1994). IEEE Computer Society Press

Wes22. Wesolowski, B.: Orientations and the supersingular endomorphism ring problem. In: Lecture Notes in Computer Science, pp. 345–371. Springer, Heidelberg, Germany (2022). https://doi.org/10.1007/978-3-031-07082-2_13