Full Quantum Equivalence of Group Action DLog and CDH, and More*

Hart Montgomery[1] ** and Mark Zhandry[2] ***

[1] Linux Foundation‡
[2] NTT Research & Princeton University

Abstract. Cryptographic group actions are a relaxation of standard cryptographic groups that have less structure. This lack of structure allows them to be plausibly quantum resistant despite Shor's algorithm, while still having a number of applications. The most famous example of group actions are built from isogenies on elliptic curves.
Our main result is that CDH for abelian group actions is quantumly *equivalent* to discrete log. Galbraith et al. (Mathematical Cryptology) previously showed *perfectly* solving CDH to be equivalent to discrete log quantumly; our result works for any non-negligible advantage. We also explore several other questions about group action and isogeny protocols.

Proving the equivalence of breaking the Diffie-Hellman protocol and computing discrete-log is one of the oldest problems in public key cryptography.

Boneh and Lipton [BL96]

1 Introduction

Diffie-Hellman key agreement [DH76] is one of the most important protocols in cryptography. Given a generator g of a cyclic group of order p, Alice and Bob choose random $a \leftarrow \mathsf{Z}_p$ and $b \leftarrow \mathsf{Z}_p$, respectively, and exchange the values g^a and g^b. Their shared key is then $g^{ab} = (g^a)^b = (g^b)^a$.

One way to break Diffie-Hellman is to compute discrete logarithms (DLog): extract a from (g, g^a) and then compute $g^{ab} = (g^b)^a$ from Alice's message. Fortunately, computing discrete logs appears hard, and after decades of cryptanalytic effort the best classical algorithms on certain groups—multiplicative groups of finite fields and elliptic curves—have sub-exponential or exponential complexity.

The security of Diffie-Hellman key exchange, however, is potentially easier than solving DLog. Indeed, computing the shared key is equivalent to solving the computational Diffie-Hellman problem (CDH): computing g^{ab} from (g, g^a, g^b). While CDH is clearly no harder than DLog, it is not a priori obvious that the converse should hold. After all, CDH and DLog are very different problems: CDH is in essence computing multiplication $a, b \mapsto a \times b$ homomorphically on

‡ A portion of this work was done when the author was employed by Fujitsu Resarch.
* The full version of this paper is available at https://eprint.iacr.org/2022/1135.
** https://orcid.org/0000-0002-8907-5791
*** https://orcid.org/0000-0001-7071-6272

the encoded values g^a, g^b, whereas DLog is inverting the encoding. The good news is that there has been classical progress towards proving such an equivalence [den90, Mau94, MW96, BL96]. However, the *polynomial-time* equivalence of DLog and CDH in general groups without any auxiliary information still remains an important fundamental open question. As such, the hardness of CDH must simply be assumed in Diffie-Hellman key exchange, requiring a potentially much stronger assumption than the hardness of DLog.

Quantum Diffie-Hellman. Shor [Sho94] shows that DLog is easy on a quantum computer, meaning the Diffie-Hellman protocol is no longer secure. Numerous proposals have been made for replacement "post-quantum" cryptosystems. One interesting example preserving the spirit of the original Diffie-Hellman protocol is due to Couveignes [Cou06] and Rostovtsev and Stolbunov [RS06]. They propose to replace the group in Diffie-Hellman with a group *action*. Very roughly, the group action allows for a similar operation as discrete exponentiation as in Diffie-Hellman, but does *not* have an analagous operation for multiplying two group elements, as is needed by Shor's attack.

In more detail, a group action consists of a group G and a set X, together with an action $\star : G \times X \to X$ such that for any $a, b \in G$ and $x \in X$, it holds that $(ab) \star x = a \star (b \star x)$. In this setting, DLog is the task of recovering a from $(x, a \star x)$, and CDH is the task of computing $(ab) \star x$ from $(x, a \star x, b \star x)$. If we consider *abelian* and *regular*[3] group actions, we can translate Diffie-Hellman key exchange from groups to group actions by viewing Z_p as the group acting on the set $\langle g \rangle$ through discrete exponentiation: $a \star x = x^a$. DLog and CDH on the group immediately correspond to DLog and CDH on the group action. However, other group actions that do not correspond to plain groups are possible. The most notable example is isogenies over elliptic curves [CLM+18], one of the leading candidates for post-quantum public key cryptography proposed by Couveignes, Rostovtsev, and Stolbunov[4]. In the full version of the paper, we discuss how other plausibly post-quantum proposals can sometimes also be phrased as group actions.

As in the classical case, the DLog-CDH equivalence is an important fundamental question in the quantum world. It may even be *more* important than the classical equivalence today, as the post-quantum hardness of group actions has so far seen a much smaller cryptanalytic effort than the classical hardness of groups, and therefore our confidence in the post-quantum CDH assumption on group actions is much weaker. An equivalence to DLog would therefore be an important step toward improving this confidence. In ordinary groups, the post-quantum equivalence is trivial: they are both easy. In group actions, however, it is less clear: group actions have less exploitable structure for proving such an

[3] A regular group action is a group action that, for every $x_1, x_2 \in X$, there exists a *unique* element $g \in G$ such that $x_1 = g \star x_2$.

[4] A few very recent works [CD22, MM22, Rob22] break a certain isogeny-based protocol called SIDH. SIDH, however, is just one of a number of isogeny protocols, and in particular it is *not* a group action. For a slightly more in depth discussion about different isogeny protocols, see Section 2.5.

equivalence, but quantum algorithms are more powerful and can potentially be used to facilitate a reduction.

In a short paper, Galbraith *et al.* [GPSV18] give a promising first step toward proving an equivalence: they show that any *perfect* algorithm for solving CDH in abelian group actions can be converted into a DLog algorithm. The core idea is that a perfect, efficient CDH algorithm essentially turns the set of a group action into a plain group, with $x_1 \times x_2 = \mathsf{CDH}(x_1, x_2)$. One can then apply Shor's DLog algorithm to the derived group. The main difficulty is that solving DLog in the derived group is not exactly identical to DLog in the original group action. Galbraith *et al.* essentially show how to translate one DLog to the other to complete the reduction.

Unfortunately, if the CDH algorithm has even relatively minor correctness error (even, say, 10%), the above algorithm does not work. On the other hand, for cryptographic applications, we want to justify that no efficient algorithm can solve CDH with any *non-negligible* success probability. It could therefore be, for example, that CDH can be broken—and hence also group action key agreement—with probability 0.9, but that DLog is still hard. In plain groups, one can amplify success probability using standard random self-reductions for CDH. However, as pointed out by Galbraith et al., the limited structure of group actions prevents such random self-reductions. They therefore leave the full quantum equivalence of DLog and CDH for group actions as an important open question.

1.1 This Work: Full Quantum Equivalence of DLog and CDH

In this work, we resolve the open question above, showing that DLog and CDH are quantumly equivalent for abelian group actions (Section 3). Since the most commonly used group actions in cryptography (from isogenies) are abelian, our results here have wide applicability and can be used directly on isogeny-based cryptosystems such as CSI-FiSh [BKV19][5].

As a secondary result, we also show that the same cannot hold generically for Decisional Diffie-Hellman (DDH), which is equivalent to asking that the shared key not only cannot be predicted by the adversary, but that it is indistinguishable from a random string. In other words, there is no black box quantum equivalence between DLog (or even CDH) and DDH (Section 4). We also formally specify a generic model for group actions (Section 5), explore relaxations of group actions relevant to certain isogeny protocols (Section 6), and discuss the relationship between group actions and the dihedral hidden subgroup problem (Section 7).

Our reduction (Section 3). Our DLog-CDH equivalence will use Galbraith et al. to reduce the problem of proving equivalence to that of boosting the success probability of a CDH algorithm. However, this comes with many challenges, which we now explore. Consider a deterministic algorithm A such that:

$$\Pr_{a,b \leftarrow G}[A(x, a \star x, b \star x) = (ab) \star x] = p \qquad \Pr_{a,b \leftarrow G}[A(x, a \star x, b \star x) = (uab) \star x] = 1 - p$$

[5] We note that our result does not directly apply to *restricted* effective group actions (REGAs) like CSIDH [CLM+18] and explain this in more detail later.

for some constant $p \in [0,1]$ and fixed known group element $u \in G \setminus \{1\}$. This would certainly be a valid CDH adversary with success probability p.

Remark 1. Throughout, we will consider x as being fixed; this is usually how CDH is modeled, and typically makes hardness results for CDH more challenging. It is also possible to consider a variant where x is chosen randomly and A works for a random x. [BMZ19] explore the fixed vs random question for plain groups.

In the plain group setting, the equivalent setup would be that A on input (g, g^a, g^b), outputs g^{ab} with probability p and g^{uab} with probability $1-p$. An easy random self-reduction for this A would be to run $h \leftarrow A(g, (g^a) \times g^c, (g^b) \times g^d)$ for random choices of c, d. Each trial will run A on random independent inputs, so we know that $h = g^{(a+c)(b+d)}$ with probability p, and $h = g^{u(a+c)(b+d)}$ with probability $1-p$. We can then compute $h' = h \times (g^a)^{-d}(g^b)^{-c}g^{-cd}$. If $h = g^{(a+c)(b+d)}$, then $h' = g^{ab}$. Meanwhile, if $h = g^{u(a+c)(b+d)}$, then $h' = g^{(u-1)(a+c)(b+d)+ab}$, which is a uniformly random element. Therefore, by repeating this process many times on independent c, d, a p fraction of the elements will be identical to g^{ab}, and the rest will be uniformly random. Taking a majority therefore gives g^{ab} with overwhelming probability. An important feature of this self-reduction is that when A is correct, the self-reduction gives the correct answer, and when A is incorrect, the self-reduction gives a uniformly random answer. The self-reduction can be strengthened to handle arbitrary A, thus giving a generic way to boost success probability.

Unfortunately, the above re-randomization is not possible with group actions, since there is no multiplication analog for set elements. Given $(x, a \star x, b \star x)$, one could try choosing a random c, d and running $(cd)^{-1} \star A(x, c \star (a \star x), d \star (b \star x))$. The result will be $(cd)^{-1} \star [(ac)(bd)] \star x = (ab) \star x$ with probability p and $(uab) \star x$ with probability $1-p$. This allows us to obtain many samples of each. But unlike the plain group self-reduction, now when A is incorrect we do not output a uniformly random answer, but instead output a fixed incorrect answer $(uab) \star x$. This means we cannot in general take a majority since if $p < 1/2$ this would actually give the incorrect answer. In this case, if we *knew* that $p < 1/2$, we would know to actually take the minority element as output. This would require making non-black box use of A, which is non-standard but acceptable. However, if $p = 1/2$, then the majority or minority element is just a random sample between $(ab) \star x$ and $(uab) \star x$. In this case, even knowing p is not enough to identify the correct answer.

We will now show how to resolve the reduction for this particular class of adversaries. To do so, we consider two cases: $u^2 = 1$, or not. The exponent 2 in $u^2 = 1$ is a result of our algorithm A outputting a random choice amongst two elements, and in more general settings we could consider higher, but still polynomial, exponents. Note that group actions are defined and plausibly hard for non-cyclic or non-prime order groups, so it is reasonable to consider group orders that have small factors. For isogenies, the group order is indeed smooth.

If $u^2 = 1$ and $p = 1/2$, we are basically stuck: A is simply outputting a random sample in the orbit of $(ab) \star x$ under action by u. Nothing we can do will amplify the success probability. Instead, we observe that A can be viewed as essentially solving CDH—with perfect probability!—in the subgroup $G/\langle u \rangle$. We then apply Galbraith et al. to this subgroup to solve DLog relative to $G/\langle u \rangle$. We can then solve for the full DLog in G by brute forcing the $\langle u \rangle$ component. This works regardless of p, but requires u to generate a small group.

If $u^2 \neq 1$ and/or if $p \neq 1/2$, another approach will work. Here, we can first run our re-randomized A several times on $(x, a \star x, b \star x)$ to obtain $y_0 = (ab) \star x$ and $y_1 = (uab) \star x$, but we do not yet know which is which. But in this case, we can use the fact that A is *not* generating uniform outputs in the orbit of $(ab) \star x$ to distinguish the two cases. Concretely, we run the re-randomized A several times on (x, x, y_0) and (x, x, y_1). Since $x = 1 \star x$, we know that (x, x, y_0) will output y_0 with probability p and $u \star y_0 = y_1$ with probability $1 - p$. This distribution of outputs exactly matches the distribution from our original set of trials on $(x, a{\star}x, b{\star}x)$. Meanwhile, (x, x, y_1) will output y_1 and $u{\star}y_1 = (u^2ab){\star}x$ with probabilities p and $1-p$. This distribution will be different than that from our original set of trials. Thus by comparing the distributions generated from (x, x, y_0) and (x, x, y_1) with the distribution generated from $(x, a \star x, b \star x)$, we can identify which of y_0, y_1 are the correct CDH output.

Our result generalizes the approach above to work with arbitrary adversaries A, and to work without needing any side-information (like the probability p) about the distribution of outputs of A. Essentially, we show that there is always a polynomial-sized subgroup H of G such that we can amplify A to have near-perfect success probability on G/H. We then apply Galbraith et al. to the subgroup, and then brute-force the quotient group.

There are a number of challenges to getting this sketch to work. One issue is to actually identify the subgroup of G. Suppose G has order $n = 2 \times 3 \times 5 \times ....$ Then the number of subgroups of polynomial-size will be $\lambda^{O \log(\lambda)}$; if G is non-cyclic, the number of small subgroups can even be exponential. So we cannot simply guess the subgroup, and must instead compute it.

Another issue is *thresholding*: we need to make decisions about whether various distributions of elements are close or far. These decisions are made by sampling a number of samples from the distributions, and comparing frequencies. But we can only obtain frequency estimates with inverse-polynomial error. For whatever criteria we use to distinguish distributions, if two distributions are close but not too close, the noise in our estimates will cause the criteria to output just a random bit. The question is then: if the various decisions underlying our algorithm may have random answers, how can we guarantee consistent outputs, as required to achieve a high success probability?

The randomness from thresholding seems impossible to fully overcome. However, we show via careful arguments that the randomness can all be contained within the choice of the subgroup H. Once this subgroup is fixed, we show that we can set our decision-making criteria such that we always make consistent decisions, resulting in consistent CDH solutions.

We note that our main proof assumes the group action is regular, meaning for a fixed x, $a \star x$ is a bijection. This is the most relevant setting to isogeny-based group actions. Nevertheless, we explain in Section 3.1 how to extend to arbitrary abelian group actions.

Impossibility of Extending to DDH (Section 4). Given the above, one may hope to actually prove that DLog implies DDH, namely that $(ab) \star x$ is indistinguishable from $c \star x$ for a random c, given $x, a \star x, b \star x$.

Unfortunately, we refute this possibility, at least in the composite-order setting that is most relevant to post-quantum cryptosystems. The idea is simple: we start with any group action $\star : G \times X \to X$ where CDH—and maybe even DDH—is hard. We then define a slightly larger group and set $G' = G \times \mathsf{Z}_p$ and $X' = X \times \mathsf{Z}_p$, for some polynomially bounded p. We expand $\star$ to an action of G' on X' by defining $(a, u) \star (x, y) = (a \star x, u + y)$. DLog and CDH easily hold for the expanded group action, but DDH is trivially false just by looking at the Z_p component, which has no hardness. We note that if G is cyclic, we can make G' cyclic as well by choosing p to be relatively prime to the order of G.

Generic Group Actions (Section 5). Next, we propose a generic group action model, analogous to the generic group model of [Sho97]. In this model, the set elements X are just random strings, and the action of G on X is provided by an oracle which can be queried by the adversary. This model is implicit in much of the prior work on group actions, but we are not aware of it being formally written down. We also note that the model trivially extends to the quantum setting, where classical queries are replaced by quantum queries.

On REGAs (Section 6). Many isogeny protocols cannot be phrased as clean group actions. Essentially, in some isogeny-based protocols (such as CSIDH [CLM+18]) there is a set of generators $g_1, \ldots, g_\ell \in G$, and it is only known how to efficiently compute the actions of the g_i or g_i^{-1}; one can then compute the action of any $g \in G$ provided one has a representation of $g = \prod_{i=1}^{\ell} g_i^{\alpha_i}$ for polynomially-sized α_i. In general, finding such a representation is believed to be hard. This setting is referred to as a *Restricted Effective Group Action (REGA)*.

Our reduction (as with Galbraith et al.) does not apply to REGAs, since applying Shor's algorithm requires the ability to compute the action of arbitrary group elements g. Formalizing some discussion from Galbraith *et al.*, we show that the reduction works for REGAs if a problem similar to the 1D Short Integer Solution (1D-SIS) problem is easy which we call REGA-SIS.[6] In the case that $G = \mathbb{Z}_p$–which we can assume since we are focused on abelian groups–the problem becomes essentially the one-dimensional version of the inhomogeneous SIS (ISIS) problem [BGLS19]: given a target integer $t \in \mathbb{Z}_p$ and a vector of integers $\mathbf{s} \in \mathbb{Z}_p^\ell$ defined by the REGA description, the problem is to find a vector of integers $\mathbf{v} \in [-\beta, \beta]^\ell$ such that $t = \mathbf{s} \cdot \mathbf{v}$. The only difference between REGA-SIS

[6] We defer a formal definition of this problem to the body of the paper. It is shown in [BLP+13] that 1D-SIS, for certain paramater settings, is equivalent to the "standard" LWE problem.

and what a natural definition of "1D-ISIS" would be is that the given vector of integers $\mathbf{s}$ is defined by the REGA rather than sampled randomly.

Essentially, we show that such a REGA-SIS oracle is enough to compute a representation of g in terms of the g_i, which converts the REGA into a standard group action. This shows that in a world where REGA-SIS is easy, our equivalence between DLog and CDH also holds for REGAs. It turns out that the hardness of REGA-SIS is, in fact, inherent in solving DLog on REGAs: we also show that any algorithm which solves DLog on REGAs can be used to solve this REGA-SIS problem. This result is quite interesting since it implies DLog on REGAs is at least as hard as a (not necessarily randomized, and thus maybe not hard) version of a hard lattice problem.

If we could somehow strengthen this to show that a CDH solver on REGAs must also solve REGA-SIS, then we would obtain a full quantum equivalence between DLog and CDH for REGAs. We do not know how to prove such a result, but we give some evidence that *generic* adversaries for CDH on REGAs may have to solve REGA-SIS or, for certain groups, 1D-SIS itself. More precisely, we show a reduction that *generic* adversaries for CDH on REGAs that make *classical* group and group action "queries" can solve REGA-SIS.[7] We leave formally proving this equivalence as an interesting and practically important open problem.

The Dihedral Hidden Subgroup Problem (Section 7). Childs et al. [CJS14] apply the Dihedral Hidden Subgroup Problem (DHSP) algorithm of [Kup05] to compute isogenies between elliptic curves. This is a special case of the folklore result that any algorithm for DHSP yields an algorithm for DLog on regular, abelian group actions. We prove this folklore theorem.

The DHSP is the main approach for cryptanalyzing regular, abelian group actions, and no known better general algorithm is known. However, we point out that the two are *not* trivially equivalent: group actions have significant extra structure that could potentially be used for attacks that is not exploited by the reduction to DHSP. We are not aware of this observation being explicitly mentioned previously.

We next conjecture that, nevertheless, DHSP and regular, abelian group actions are *generically* equivalent, meaning any generic algorithm for solving these group actions can be used to solve DHSP generically. We offer some evidence of this conjecture, but leave proving or disproving it as a fascinating open question.

2 Preliminaries

In this section we discuss background material that is used in the rest of the paper. We expect that experienced readers can skip this section. For a more thorough presentation of preliminary material, please see the full version of the paper.

[7] The adversary could be quantum but is restricted to classical queries to the group and group action oracles.

2.1 Min-entropy and Leftover Hash Lemma

Let Z be a discrete random variable Z with sample space Ω. Its *min-entropy* is

$$H_\infty(Z) = \min_{\omega \in \Omega}\{-\log \Pr[Z = \omega]\}.$$

For two random variables Y and Z, we use $H_\infty(Z|Y)$ to denote the min-entropy of Z conditioned on Y. We will use the following lemma, which is a simplified version of the leftover hash lemma [ILL89].

Lemma 1. *Let $\{H_s : \mathcal{Z} \to Y\}_{s \in \mathcal{S}}$ be a family of pairwise independent hash functions, and Z and S be discrete random variables over $\mathcal{Z}$ and $\mathcal{S}$, respectively. If $H_\infty(Z) > \log|Y| + 2\log(\varepsilon^{-1})$ we have $\Delta[(S, H_S(Z)), (S, U)] \leq \varepsilon$, where Δ denotes statistical distance and U denotes the uniform distribution over Y.*

We will also use the following corollary of the leftover hash lemma.

Lemma 2. *Let G be an (additive) finite abelian group such that $|G| = \lambda^{\omega(1)}$. Let $n \in Z$ such that $n > \log|G| + \omega(\log(\lambda))$. If $\mathbf{g} \leftarrow G^n$ and $\mathbf{s} \leftarrow \{0,1\}^n$, then*

$$\left(\mathbf{g}, \sum_{i=1}^{n} s_i \cdot g_i\right) \overset{s}{\approx} (\mathbf{g}, u),$$

where $u \leftarrow G$ is a uniformly chosen element from G.

2.2 1D-SIS Problem

The 1D-SIS problem dates to the original work of Ajtai [Ajt96] and has been used in many cryptographic applications [BV15, BKM17]. These cases use special moduli, but the case for general moduli follows from [BLP$^+$13], where it is shown that the 1D-SIS problem with certain parameters but no special restrictions on the modulus is as hard as standard polynomial modulus LWE.

Definition 1. *Let m, β, and q be positive integers. In the $1D\text{-}SIS_{m,q,\beta}$ problem, an adversary is given a random vector $\mathbf{v} \leftarrow \mathbb{Z}_q^m$ and asked to provide a vector $\mathbf{u} \in \mathbb{Z}_q^m$ such that $||\mathbf{u}|| < \beta$. We say that an adversary efficiently solves the $1d\text{-}SIS_{m,q,\beta}$ problem if it can provide such a vector in PPT time.*

2.3 Cryptographic Group Actions

Here we define cryptographic group actions following Alamati *et al.* [ADMP20], which are based on those of Brassard and Yung [BY91] and Couveignes [Cou06].

Definition 2. (Group Action) *A group G is said to* act on *a set X if there is a map $\star : G \times X \to X$ that satisfies the following two properties:*

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

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

Definition 3. *A group action* $(G, X, \star)$ *is said to be* regular *if, for every* $x_1, x_2 \in X$*, 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 $x \in X$, the map $f_x : g \mapsto g \star x$ defines a bijection between G and X; in particular, if G (or X) is finite, then we must have $|G| = |X|$.

In this paper, unless we specify otherwise, we will work with *effective* group actions (EGAs). An effective group action $(G, X, \star)$ 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 (see the full version of our paper and our discussion on KEGAs for more details), and all of the less obvious properties needed for EGAs follow automatically. However, the definition of an EGA itself is a little bit tedious (and quite formal so as to properly model isogeny-based constructions in a classical world) so we defer it to the full version of the paper.

2.4 Computational Problems

We next define problems related to group action security that are more semantically similar to typical group-based problems than those that are traditionally used in isogeny litaterature. We define the formal definitions that are typically used in isogenies (based on [ADMP20] in the full version of the paper, where we also compare them to our (intuitively simpler, but almost equivalent) notions of security defined here. We emphasize that we are defining *problems* here and not *assumptions* because these are easier to use in reductions.

Definition 4. (Group Action Discrete Logarithm) *Given a group action* $(G, X, \star)$ *and distributions* $(\mathcal{D}_X, \mathcal{D}_G)$*, the group action discrete logarithm problem is defined as follows: sample* $g \leftarrow \mathcal{D}_G$ *and* $x \leftarrow \mathcal{D}_X$*, 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* $(G, X, \star)$ *and distributions* $(\mathcal{D}_X, \mathcal{D}_G)$*, the group action CDH problem is defined as follows: sample* $g \leftarrow \mathcal{D}_G$ *and* $x, x' \leftarrow \mathcal{D}_X$*, 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* $y' = g \star x'$*.*

Definition 6. (Group Action Decisional Diffie-Hellman (DDH)) *Given a group action $(G, X, \star)$ and distributions $(\mathcal{D}_X, \mathcal{D}_G)$, the group action DDH problem is defined as follows: sample $g_1, g_2 \leftarrow \mathcal{D}_G$ and $x, z' \leftarrow \mathcal{D}_X$, compute $y_1 = g_1 \star x$, $y_2 = g_2 \star x$, and $z = g_1 g_2 \star x$.*

The group action DDH problem is to distinguish whether a tuple is of the form (x, y_1, y_2, z) or (x, y_1, y_2, z').

Remark 2. The above definitions allow for different distributions $\mathcal{D}_X$ on X. In particular, $\mathcal{D}_X$ could be uniform over 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, our reduction between DLog and CDH will preserve x, and therefore it works no matter how x is modeled.

2.5 Instantiations of Cryptographic Group Actions

We next discuss various instantiations of cryptographic group actions and where they fall into our definitions. We start by discussing isogenies. For more details, we refer the reader to [ADMP20], which has an extensive discussion on the classification of various isogeny protocols into group action definitions.

Isogenies that are EGAs. CSI-FiSh [BKV19] and its derivatives/applications [DM20a] have EGA functionality and are conjectured to even have weak pseudorandomness. However, there have recently been some subexponential attacks on CSI-FiSh [Pei20, BS20] and current cryptosystems built from CSI-FiSh are not particularly efficient. In fact, there are not efficient algorithms to (asymptotically) generate parameter sets for CSI-FiSh. However, if a powerful quantum computer were available, then efficient (quantum) computation of the class group structure could be used to generate arbitrary parameter sets for CSI-FiSh and improve efficiency.

Isogenies that are *restricted* EGAs (REGAs). Recall that, in a *REGA*, there is a set of generators $g_1, \ldots, g_\ell \in G$, and it is only known how to efficiently compute the actions of the g_i or g_i^{-1}; one can then compute the action of any $g \in G$ provided one has a representation of $g = \prod_{i=1}^{\ell} g_i^{\alpha_i}$ for polynomial α_i. We define REGAs formally in the full version of the paper. Many of the most commonly used isogeny protocols are based on CSIDH [CLM+18], which is a REGA. These include things like the signature scheme SeaSign [DG19] or OT protocols [LGdSG21].

Isogenies that are not GAs. There are many isogeny-based schemes that cannot be modeled as group actions. Examples include SIDH [DJP14] and the recently proposed OSIDH [CK20, Onu21, DDF21]. Most isogeny-based protocols that are not group actions are typically used for key exchange or other very simple cryptographic applications.

Remark 3. A few very recent works [CD22, MM22, Rob22] break SIDH by showing how to solve the discrete log problem. However, the attack crucially exploits certain extra points that are made public in SIDH, and these points are precisely one of the reasons that SIDH is not a group action. In particular, the the attack does not seem to apply to CSI-FISH or CSIDH, the main instantiations of EGAs and REGAs, respectively.

Non-Isogeny Group Actions. Currently all instantiations of *abelian* candidate cryptographic group actions that are thought to be secure are isogeny-based [DDF21]. There have been a number of attempts to build key exchange and other basic primitives from nonabelian groups that amount to group actions or have hardness assumptions that can be modeled in some way as group actions [KLC+00, Sti05, SU05a], but the proposed instantiations of these schemes have been completely cryptanalyzed [Shp08, BKT18].

We note that these candidate cryptosystems typically propose an abstract scheme and then attempt to instantiate it with a group. We note that it is not usually the case that the abstract schemes themselves are broken: the cryptanalysis typically works directly on the instantiations, so it is possible that some of these protocols could be implemented securely with different choices of groups.

There have also been some candidate nonabelian cryptographic group actions proposed [JQSY19]. While these are not known to be insecure, they have far fewer applications than abelian group actions.

3 Reducing DLog to CDH Quantumly

Let $(G, X, \star)$ be a regular abelian group action. In Section 3.1 we explain how to extend our reduction to non-regular abelian actions. Let $x \in X$ be a fixed set element.

Theorem 1. *If DLog is post-quantum hard in $(G, X, \star)$, then so is CDH. More precisely, there exists an oracle algorithm $R^{\mathcal{A},(G,X,\star)}(\mu, y)$ that runs in time poly$(1/\mu, \log |G|)$ and makes poly$(1/\mu, \log |G|)$ total queries to a supposed CDH adversary $\mathcal{A}$ and group action $(G, X, \star)$, such that the following holds. If $\Pr_{a,b \leftarrow G}[\mathcal{A}(a \star x, b \star x) = (ab) \star x] \geq \mu$, then for any $a \in G$, $\Pr[R^{\mathcal{A},(G,X,\star)}(\mu, a \star x) = a] \geq 0.99$.*

We note that the above means that R is very slightly non-black box, in that its running time and number of calls to $\mathcal{A}$ depend on the success probability μ of $\mathcal{A}$. We note that any amplification of success probability (say, from μ to 0.99) will always come with such a dependence on μ. In our case, amplification is critical to our algorithm, and the dependence on μ would persist even if we only wanted $R^{\mathcal{A}}$ to have very small success probability. The remainder of this section is devoted to proving Theorem 1.

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 will also allow CDH to take as input a vector of elements, behaving as $\mathsf{CDH}(a_1 \star x, \cdots, a_n \star x) = (a_1 \cdots a_n) \star x$. Furthermore, we will allow CDH to take as input distribution(s) over the set X; in this case, CDH will also output a distribution.

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

$$q := \Pr[\mathcal{A}(y, z) = \mathsf{CDH}(y, z)]$$

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 introduce a random self-reduction for CDH. In the case of groups (as opposed to group actions), a more powerful random self-reduction allows for amplifying the success probability on any input. The result would be an algorithm for CDH with overwhelming success probability. In our case, due to the restricted nature of group actions, we can only perform a more limited self-reduction. Nevertheless, this self-reduction has useful properties.

The Basic Random Self-reduction. The random self-reduced version of $\mathcal{A}$, denoted $\mathcal{A}_0$, works as follows:

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

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

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

which is the correct output for CDH. Moreover, the set elements y', z' are uniformly distributed over the possible set elements.

Let $\mathcal{D}$ be the distribution $\mathcal{A}_0(x, x)$. That is, we are feeding the "dummy" distribution to our random self-reduction. While we know what the answer should be ($x = \mathsf{CDH}(x, x)$), we use this distribution to learn more about $\mathcal{A}$'s behavior.

Lemma 3. $\Pr[x \leftarrow \mathcal{D}] = q$.

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

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 $D_{y,z}$ be the distribution of outputs of $\mathcal{A}_0(y, z)$.

Lemma 4. *For every $y, z \in X$ 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. □

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 4 shows that $\mathcal{D}_{y,z}$ outputs $\mathsf{CDH}(y, z)$ with probability q. 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 of the list of outputs is the correct answer; we cannot determine this yet.

In the following, we will take steps to remedy this issue. Throughout this section, it is instructive to keep the following examples in mind:

1. Let $g \in G \setminus \{1\}$. $\mathcal{A}(Y, Z)$ outputs $\mathsf{CDH}(y, z)$ with probability $1/3$, and $g \star \mathsf{CDH}(Y, Z)$ with probability $2/3$. Notice that in this case, $\mathcal{A}_0$ has the same distribution of outputs as $\mathcal{A}$. Also notice that taking the majority element will give the wrong answer. Thus, we cannot immediately decide which of the outputs of $\mathcal{A}_0$ is the right answer just by looking at the frequencies.
2. Let $\mathcal{H}$ be a subgroup of G of size $1/q$. Then consider the case where $\mathcal{A}(y, z)$ outputs $c \star \mathsf{CDH}(y, z)$, where $c \leftarrow \mathcal{H}$ is chosen uniformly. Note that $\mathcal{A}$ is still correct with probability q in this case, since $c = 1_{\mathcal{H}}$ with probability q. Similar to Example 1, there is no way to identify the correct output just by looking at frequencies.
3. Suppose $\mathcal{H} = \mathsf{Z}_2^{\log \lambda}$, which we can decompose as a chain of subgroups $\mathcal{H}_i = \mathsf{Z}_2^i$ with $\mathcal{H}_{i-1} \subseteq \mathcal{H}_i$. $\mathcal{A}$ outputs $c \star \mathsf{CDH}(y, z)$, where $c \in \mathcal{H}$. However, c is not uniform. Instead, $i \in [0, \log \lambda]$ is chosen according to some probability distribution, and then c is chosen uniformly from $\mathcal{H}_i$.
4. Suppose $\mathcal{H} = \mathsf{Z}_2^{\log \lambda}$. Again, $\mathcal{A}$ outputs $c \star \mathsf{CDH}(y, z)$, where $c \in \mathcal{H}$ but not uniform. Here, c occurs with probability $1 - \alpha|c|_1$, where $|c|_1$ denotes the Hamming weight of c.

Example 1. It turns out Example 1 can be handled using the shifting property from Lemma 4. Suppose we are given a CDH challenge parameterized by $(y = a \star x, z = b \star x)$. Basically, after repeating many runs of $\mathcal{A}_0\,(y, z)$, we obtain two elements: $w_0 = (ab) \star x$ and $w_1 = (gab) \star x$. In theory, in this example we could exploit the fact that we know the probabilities with which $\mathcal{A}$ outputs the correct set element and the "g-multiplied" set element, but let's assume that we do not know this. What can we do?

Suppose we feed these outputs back into $\mathcal{A}_0$, running $\mathcal{A}_0\,(w_0, x)$ and $\mathcal{A}_0\,(w_1, x)$ several times each. Each of these two runs will output two distinct elements. Since

$w_0 = (ab) \star x$, Lemma 4 shows that $\mathcal{A}_0(w_0, x) = \mathcal{D}_{w_0} = \mathcal{D}_{y,z} = \mathcal{A}_0(y, z)$ as distributions. Likewise, since $w_1 = (gab) \star x$, we have $\mathcal{A}_0(w_1, x) = \mathcal{A}_0(g \star y, z)$.

Therefore, because $\mathcal{A}_0(w_0, x)$ is distributed the same as $\mathcal{A}_0(y, z)$ and $\mathcal{A}_0(w_1, x)$ is not, we can effectively distinguish w_0 from w_1 and find the correct CDH output.

Example 2. On the other hand, Example 2 is much harder to handle. Mimicking the above, we first run $\mathcal{A}_0$ several times, obtaining the list of values $c \star \mathsf{CDH}(y, z)$ as c ranges over $\mathcal{H}$, but we don't know c. We can then try, for each $c \star \mathsf{CDH}(y, z)$, running $\mathcal{A}_0(c \star \mathsf{CDH}(y, z), x)$ several times, to obtain tuples of elements. However, this will not give us any useful information: each tuple will be exactly the same list as in the original run of $\mathcal{A}_0$, namely the entire set $\mathcal{H} \star \mathsf{CDH}(y, z)$. The problem is that the output distribution of $\mathcal{A}_0$ is invariant under action by $\mathcal{H}$.

Looking ahead, we cannot improve the CDH algorithm for this example. However, this particular example gives a *perfect* CDH oracle relative to the group $G/\mathcal{H}$ acting on $X/\mathcal{H} := \{\mathcal{H} \star w : w \in X\}$. We will use such an algorithm to solve discrete log in $G/\mathcal{H}$. We can then solve discrete logarithms in $\mathcal{H}$ by brute force, and then piece the two results together to solve discrete logarithms in G.

Examples 3 and 4. In general, however, we may not get a perfect CDH oracle for $\mathcal{H}$, and are not even obviously guaranteed that the outputs lie in a small subgroup. In Example 3, consider the distribution over i such that larger subgroups are very unlikely, but not *too* unlikely. For any fixed number of queries, it could be that, with probability $1/2$, all results end up in $\mathcal{H}_i$, but with probability $1/2$ some of the results will end up in $\mathcal{H}_{i+1}$. It might, a priori, not even be possible to identify when you have all the elements from a subgroup, since "chaining" calls to $\mathcal{A}_0$ as we have done above might move us outside a subgroup. So it is unclear if there is a way to always output a consistent complete subgroup, so as to get a near-perfect CDH solver relative to G mod this subgroup.

Next, we will gradually improve our CDH solver to resolve these difficulties.

Restricting to a small subgroup. We show how to discard some wrong outputs of $\mathcal{A}_0$ so that the remaining outputs lie in a reasonably-small subgroup of G, while still guaranteeing that we keep $\mathsf{CDH}(y, z)$.

We first give some notation. For any two distributions $\mathcal{D}_0, \mathcal{D}_1$ over X, let $\|\mathcal{D}_0 - \mathcal{D}_1\|_\infty = \max_{w \in X} |\Pr[w \leftarrow \mathcal{D}_0] - \Pr[w \leftarrow \mathcal{D}_1]|$. For a distribution $\mathcal{D}$ over X, consider sampling T elements $w_1, \ldots, w_T$ from $\mathcal{D}$. This vector of w_i gives rise to an "empirical" distribution $\tilde{\mathcal{D}}$, where the probability of any w is just the relative frequency of w amongst the w_i. Note that even though $\tilde{\mathcal{D}}$ has a domain of exponential size, we can represent it by the list $w_1, \ldots, w_T$, which has size T. Also note that there are two distributions here: the empirical distribution $\tilde{\mathcal{D}}$ itself, and the distribution over empirical distributions. We denote the latter as $\tilde{\mathcal{D}} \leftarrow \mathcal{D}^T$.

We are now ready to give our next algorithm, $\mathcal{A}_1(y, z)$:

– Let $T = \lambda/\delta^2$ for some parameter $\delta \in (0, 1)$.

– Run $\tilde{\mathcal{D}}^* \leftarrow \mathcal{A}_0(y,z)^T$
– For each w in the support of $\tilde{\mathcal{D}}^*$, run $\tilde{\mathcal{D}}_w \leftarrow \mathcal{A}_0(w,x)^T$.
– Output L, the set of w in the support of $\tilde{\mathcal{D}}^*$ such that $\|\tilde{\mathcal{D}}_w - \tilde{\mathcal{D}}^*\|_\infty \leq \delta/2$.

We will think of λ being $\mathrm{poly}(\log q)$, so that $2^{-\Omega(\lambda)}$ is negligible in $1/q$. Note that $\mathcal{A}_1$ makes at most $T^2+T = O(\lambda^2/\delta^4)$ evaluations of $\mathcal{A}_1$, and hence T^2+T evaluations of $\mathcal{A}_0$ and $O(T^2+T)$ group action operations. In order to analyze the algorithm $\mathcal{A}_1$, we need to give some basic results. First we recall the Dvoretzky-Kiefer-Wolfowitz inequality:

Lemma 5 ([Mas90]). *For any $\zeta > 0$ and distribution $\mathcal{D}$, except with probability $2e^{-2\zeta^2 T}$, $\|\tilde{\mathcal{D}} - \mathcal{D}\|_\infty \leq \zeta$, where $\tilde{\mathcal{D}} \leftarrow \mathcal{D}^T$.*

In other words, the empirical distribution converges to the underlying distribution $\mathcal{D}$ as the number of samples T grows large.

Now consider the distribution $\mathcal{D} = \mathcal{A}_0(x,x)$ from before, and the derived distributions $\mathcal{D}_w = \mathsf{CDH}(w,\mathcal{D})$. Let $d_w = \|\mathcal{D}_w - \mathcal{D}\|_\infty$.

Lemma 6. *$\forall y,z \in X$, $\|\mathcal{D}_{\mathsf{CDH}(y,z)} - \mathcal{D}_y\|_\infty = d_z$ and $d_{\mathsf{CDH}(y,z)} \leq d_y + d_z$.*

Proof. For the equality, note that $\|\mathcal{D}_{\mathsf{CDH}(y,z)} - \mathcal{D}_y\|_\infty = \|\mathsf{CDH}(y,\mathcal{D}_z) - \mathsf{CDH}(y,\mathcal{D})\|_\infty$. Since $\mathsf{CDH}(y,\cdot)$ simply permutes the elements of X—more precisely, it maps $v \in X$ to $a \star v$ where $y = a \star x$—it does not affect the distance between distributions, and therefore $|\mathsf{CDH}(y,\mathcal{D}_z) - \mathsf{CDH}(y,\mathcal{D})| = |\mathcal{D}_z - \mathcal{D}| = d_z$. For the inequality, we have $d_{\mathsf{CDH}(y,z)} = |\mathcal{D}_{\mathsf{CDH}(y,z)} - \mathcal{D}|_\infty \leq |\mathcal{D}_{\mathsf{CDH}(y,z)} - \mathcal{D}_y|_\infty + |\mathcal{D}_y - \mathcal{D}|_\infty = d_z + d_y$, where we used the equality in the second to last step. □

Now we prove the following general result about abelian groups. Fix an abelian group $\mathcal{H}$ and a set of generators $\mathbf{a} = (a_1, \ldots, a_n)$. For any vector $\mathbf{e} \in \mathbb{N}^n$ of non-negative integers, define $\mathbf{a}^{\mathbf{e}} := \prod_{i=1}^n a_i^{e_i}$. Let $\|\mathbf{e}\|_1 := \sum_{i=1}^n |e_i|$. Then for any $r \in \mathcal{H}$, we define $\|r\| := \min_{\mathbf{e} \in \mathbb{N}^n : r = \mathbf{a}^{\mathbf{e}}} \|\mathbf{e}\|_1$.

Lemma 7. *If $U = \{r \in \mathcal{H} : \|r\| \leq ns\}$ has size at most s, then $U = \mathcal{H}$.*

In other words, if the subset of $\mathcal{H}$ with small $\|\cdot\|$ is not too big, then in fact all of $\mathcal{H}$ has small $\|\cdot\|$.
Proof. Clearly $U \subseteq H$. In the other direction, consider a single a_i. Since U has size at most s, then so does the set $\{a_i^{e_i} : 0 \leq e_i \leq s\} \subseteq U$. As there are $s+1$ different possibilities for e_i, there must be $e_i' < e_i$ such that $a_i^{e_i} = a_i^{e_i'}$. Then $a_i^{e_i' - e_i} = 1$, and $0 < e_i' - e_i \leq s$. For any $r \in \mathcal{H}$, write $r = \mathbf{a}^{\mathbf{e}}$. Since a_i has order at most s, we can reduce each e_i to an integer smaller than s without changing r. After such a reduction, $\|\mathbf{e}\|_1 \leq ns$, and so $r \in U$. Hence $H \subseteq U$. □

Let $L_\delta \subset G$ be the set of all $a \in G$ such that $d_{a \star x} \leq \delta$, and $\mathcal{H}_\delta$ be the subgroup of G generated by L_δ. We have the following:

Lemma 8. *Let $\epsilon \in (0,1]$ be a real number. Then if $\delta \leq \epsilon q^4/8$, $|\mathcal{H}_\delta| \leq q^{-1} + \epsilon$.*

Note that ϵ is necessary: $\mathcal{D}$ may output $g \star x$ for a g in a subgroup $\mathcal{H}$ of size n, with q^{-1} negligibly smaller than n. Suppose $\Pr[x \leftarrow D] = q$ and $\Pr[g \star x \leftarrow D]$ is slightly less than q for all other g. Then $\mathcal{H}_\delta = \mathcal{H}$ for any non-negligible δ.

Proof. We first prove that $|L_\delta| \leq q^{-1} + \epsilon$. Note that $d_{1_G \star x} = d_x = 0$ and so $1_G \in L_\delta$. From Lemma 3, $\Pr[x \leftarrow \mathcal{D}] = q$. Therefore, for any $a \in L_\delta$,

$$\Pr[a^{-1} \star x \leftarrow \mathcal{D}] = \Pr[x \leftarrow \mathcal{D}_{a \star x}] \geq \Pr[x \leftarrow \mathcal{D}] - \delta = q - \delta,$$

where the inequality follows since $d_{a \star x} \leq \delta$ for $a \in L_\delta$. Then

$$\begin{aligned} 1 &= \sum_{a \in G} \Pr[a^{-1} \star x \leftarrow \mathcal{D}] \geq \sum_{a \in L_\delta} \Pr[a^{-1} \star x \leftarrow \mathcal{D}] \\ &= \Pr[1 \star x \leftarrow \mathcal{D}] + \sum_{a \in L_\delta \setminus \{1\}} \Pr[a^{-1} \star x \leftarrow \mathcal{D}] \geq q + (|L_\delta| - 1)(q - \delta) \end{aligned}$$

Solving for $|L_\delta|$ gives $|L_\delta| \leq (1-\delta)/(q-\delta)$. Setting the right hand side to be $\leq q^{-1} + \epsilon$ gives the desired bound whenever $\delta \leq \epsilon q^2/(1 - q + q\epsilon)$. Note that $(1 - q + q\epsilon) \leq 1$. Therefore, $\delta \leq \epsilon q^4/8$ is only a stronger bound on δ.

We now bound $|\mathcal{H}_\delta|$ by applying Lemma 7 to $\mathcal{H} = \mathcal{H}_\delta$ and $\mathbf{a} = L_\delta$ and $s = 1/q + \epsilon$. Consider some $r = \mathbf{a}^{\mathbf{e}}$ in $\mathcal{H}_\delta$. Then by iteratively applying Lemma 6,

$$d_{r \star x} = d_{\mathsf{CDH}(\underbrace{a_1 \star x, \cdots, a_1 \star x}_{e_1}, \underbrace{a_2 \star x, \cdots, a_2 \star x}_{e_2}, a_3 \star x, \cdots)} \leq \sum_i e_i d_{a_i \star x} \leq \sum_i e_i \delta = |\mathbf{e}|_1 \delta$$

By minimizing over all $\mathbf{e}$, we have that $d_{r \star x} \leq \|r\|\delta$. For U as in Lemma 7, this means that $\Pr[r^{-1} \star x \leftarrow \mathcal{D}] = \Pr[x \leftarrow \mathcal{D}_{r \star x}] \geq q - \|r\|\delta \geq q - ns\delta$. Since the probabilities of each outcome sum to at most 1, we therefore have that $|U| \leq (q - ns\delta)^{-1}$. In order to satisfy the conditions of Lemma 7, we therefore need $1/(q - ns\delta) \leq s$, which is equivalent to $1 \leq s(q - ns\delta)$. Since $n = |L_\delta| \leq 1/q + \epsilon$, we have that this inequality is satisfied whenever $\delta \leq \epsilon q^4/(1+\epsilon q)^3$. As $1 + \epsilon q \leq 2$, our bound of $\delta \leq \epsilon q^4/8$ is only a stronger bound, showing that $\mathcal{H}_\delta = L_\delta$. Our prior bound on $|L_\delta|$ thus proves Lemma 8. ∎

We are finally ready to analyze the algorithm $\mathcal{A}_1$. Let $\mathcal{D}'$ be the distribution $\mathcal{A}_1(x, x)$, and $\mathcal{D}'_{y,z}$ be the distribution $\mathcal{A}_1(y, z)$. The next lemma follows immediately from Lemma 4:

Lemma 9. *For every $y, z \in X$ where $y = a \star x$ and $z = b \star x$ for some $a, b \in G$, $\mathcal{D}'_{y,z} = \mathsf{CDH}(y, z, \mathcal{D}')$.*

Thus, we define $\mathcal{D}'_w := \mathcal{D}'_{w,1} = \mathcal{D}'_{1,w} = \mathcal{D}'_{y,z}$, if $\mathsf{CDH}(y, z) = w$. We now prove:

Lemma 10. *Except with probability $2(T+1)e^{-\delta^2 T/8} + (1-q)^T \leq 2^{-\Omega(\lambda)}$ over $L \leftarrow \mathcal{A}_1(x, x)$, we have that $x \in L \subseteq \mathcal{H}_\delta \star x$.*

Proof. Suppose we set $\zeta = \frac{\delta}{4}$. By Lemma 5, we have that $|\tilde{\mathcal{D}}^\star - \mathcal{D}_{\mathsf{CDH}(y,z)}|_\infty \leq \delta/4$ and for each w in the support of $\tilde{\mathcal{D}}^\star$, $|\tilde{\mathcal{D}}_w - \mathcal{D}_w| \leq \delta/4$, each individually except with probability at most $2e^{-\delta^2 T/8}$. We also have that with probability $1-(1-q)^T$, x will be amongst the T samples of $\mathcal{A}_1(x,x)$. By a union bound, all of these happen simultaneously, except with probability $2(T+1)e^{-\delta^2 T/8} + (1-q)^T$.

If all of these happen, then $|\tilde{\mathcal{D}}_x - \tilde{\mathcal{D}}^\star| \leq |\tilde{\mathcal{D}}_x - \mathcal{D}| + |\tilde{\mathcal{D}}^\star - \mathcal{D}| \leq 2\delta/4 = \delta/2$. Thus $x \in L$ assuming the above hold. On the other hand, for any $w \in L$, $d_w = |\mathcal{D}_w - \mathcal{D}| \leq |\tilde{\mathcal{D}}_w - \mathcal{D}_w| + |\tilde{\mathcal{D}}_w - \tilde{\mathcal{D}}^\star| + |\tilde{\mathcal{D}}^\star - \mathcal{D}| \leq \delta$. Hence $w \in L_\delta \star x$ by the definition of the set L_δ, which immediately implies that each $w \in \mathcal{H}_\delta \star x$. $\square$

As a consequence, we have that $\mathcal{D}'$ has negligible support outside of $\mathcal{H}_\delta \star x$. Note that $\mathcal{D}'$ may not be random in $\mathcal{H}_\delta \star x$, as the list L may not include all of $\mathcal{H}_\delta \star x$, and L itself may be randomized. Indeed, in Example 4, α may be such that $\mathcal{D}$ and $\mathcal{D}_{c\star w}$ are sufficiently close for c with small Hamming weight, but $\mathcal{D}_{c\star w}$ is far for c with large Hamming weight. Some c may even be right on the cusp, being included in L with constant probability. The result is that the output may not be a whole subgroup and may have entropy.

We note that by setting ϵ a constant and $\delta = \epsilon q^4/8 = O(q^4)$, we have that $\mathcal{A}_1$ runs in time $O(\lambda q^{-8}) = \tilde{O}(q^{-8})$ and makes $\tilde{O}(q^{-8})$ total queries to $\mathcal{A}$ and the group action operations.

Filling an entire subgroup. $\mathcal{A}_1$ outputs a subset of $\mathcal{H}_\delta \star \mathsf{CDH}(y,z)$, and the subset must include $\mathsf{CDH}(y,z)$. We will now devise a new algorithm $\mathcal{A}_2$ which outputs $\mathcal{H} \star \mathsf{CDH}(y,z)$, where $\mathcal{H}$ is a (potentially unknown) subgroup of $\mathcal{H}_\delta$. We split $\mathcal{A}_2(y,z)$ into two phases, $\mathcal{A}_2^0()$, which outputs the set $\mathcal{H} \star x$, and then $\mathcal{A}_2^1(y,z,\mathcal{H} \star x)$, which outputs the set $\mathcal{H} \star \mathsf{CDH}(y,z)$. We first give $\mathcal{A}_2^0()$:

- Initialize list $L = \{x\}$. Let $s = q^{-1} + \epsilon$ be an upper bound on the size of $\mathcal{H}_\delta$.
- Let $T = s\lambda/\tau$, for a parameter $\tau \in (0,1)$ to be chosen later.
- Repeat the following at least T times:
  - For each pair $(w,w') \in L^2$, run $L_{w,w'} \leftarrow \mathcal{A}_1(w,w')$
  - Let $L' = \cup_{w,w'} L_{w,w'}$
  - If $|L'| = |L|$ and the number of iterations so far is $\geq T$, terminate and output L. Otherwise (if the number of iterations is $< T$ or $|L'| \neq |L|$), replace L with L', and continue.

We now analyze the algorithm $L \leftarrow \mathcal{A}_2^0()$.

Lemma 11. *Except with negligible probability $2^{-\Omega(\lambda)}$, all of the following hold:*

- *$L = \mathcal{H} \star x$ for some (potentially unknown) subgroup $\mathcal{H} \subseteq \mathcal{H}_\delta$.*
- *$\mathcal{A}_2^0()$ will terminate in at most $T + s$ steps.*
- *For the resulting $\mathcal{H}$, $\Pr[M \not\subseteq \mathcal{H} : M \leftarrow D'] < \tau$.*

Proof. Combining Lemmas 9 and 10, we know that except with probability $2^{-\Omega(\lambda)}$, $L_{w,w'}$ will be a list containing $\mathsf{CDH}(w,w')$. Throughout the rest of the proof of Lemma 11, we will therefore assume $\mathsf{CDH}(w,w') \in L_{w,w'}$ for all iterations and for all w, w'.

We first argue that $L \subseteq L'$ in every iteration, except with probability $2^{-\Omega(\lambda)}$. In particular, since L is set to L' at the end of each iteration, this means that L

is never decreasing in size, and once an element is added to L it will remain for the rest of the algorithm. Indeed, L initially contains x. By induction, assume L contains x for the first i iterations, and consider computing L' in this iteration. L' is set to $L' = \cup_{w,w'} L_{w,w'}$ where $L_{w,w'} \leftarrow \mathcal{A}_1(w, w')$ as w, w' range over L. In particular, since $x \in L$, L' will contain $L_{w,x} \leftarrow \mathcal{A}_1(w, x)$ for every $w \in L$. Since we assume $L_{w,x}$ contains $\mathsf{CDH}(w, x) = w$, every $w \in L$ will be included in L'.

Therefore, if $|L'| = |L|$, it must mean that $L' = L$. Additionally, once we terminate, we know that $\mathsf{CDH}(w, w') \in L' = L$ for every $w, w' \in L$, meaning L is closed under CDH/multiplication once we terminate. Hence, L forms $\mathcal{H} \star x$ for some subgroup $\mathcal{H}$. By Lemma 10, our algorithm maintains the invariant that $L \subseteq \mathcal{H}_\delta$ at all times, and hence $\mathcal{H} \subseteq \mathcal{H}_\delta$.

Now consider any $w \in \mathcal{H}_\delta$ such that $\Pr[w \in M : M \leftarrow \mathcal{D}'] \geq \tau/s$. Then after T iterations, the probability w never gets added to L is $(1 - \tau/s)^T = (1 - \tau/s)^{s\lambda/\tau} \approx e^{-\lambda}$. Union bounding over at most s such w, we see that all such w get added to L, except with probability at most $2^{-\Omega(\lambda)}$. In this case, a union bound over the w such that $\Pr[w \in M : M \leftarrow \mathcal{D}'] < \tau/s$, of which there are at most s, shows that the probability of sampling any value not in $\mathcal{H}$ is less than τ. □

We now give the algorithm $\mathcal{A}_2^1(y, z, L)$:

- Initialize $\mathcal{M}$ to be an empty list of unordered sets.
- Repeat the following λ times:
  - Run $M \leftarrow \mathcal{A}_1(y, z)$.
  - For each $w \in M, w' \in L$, run $M_{w,w'} \leftarrow \mathcal{A}_1(w, w')$.
  - Let $M = \cup_{w,w'} M_{w,w'}$. Add M to $\mathcal{M}$ (keeping duplicates).
- Let M^* be the most common element in $\mathcal{M}$.

We now analyze the algorithm $\mathcal{A}_2^1(y, z, L)$.

Lemma 12. *If $\tau \leq 1/4(s^2 + 1)$, then except with probability $2^{-\Omega(\lambda)}$, $L = \mathcal{H} \star x$ for some subgroup $\mathcal{H} \subseteq \mathcal{H}_\delta$, and $M^* = \mathsf{CDH}(y, z, \mathcal{H} \star x)$.*

Proof. Define $w^* = \mathsf{CDH}(y, z)$. We assume the bullets of Lemma 11 hold, which Lemma 11 shows hold except with probability $2^{-\Omega(\lambda)}$. Therefore, $L = \mathcal{H} \star x$ for some subgroup $\mathcal{H} \subseteq \mathcal{H}_\delta$. It remains to show that $M^* = \mathsf{CDH}(y, z, \mathcal{H} \star x) = \mathcal{H} \star w^*$. By union-bounding over the $s^2 + 1$ runs of $\mathcal{A}_1$ in each iteration and invoking the last bullet of Lemma 11, the following holds: for each iteration, except with probability at most $\tau \times (s^2 + 1) \leq 1/4$, we have that

- $M_{w,w'} \subseteq \mathcal{H} \star w^*$ for each $w \in M, w' \in L$, and therefore in particular $M \subseteq \mathcal{H} \star w^*$.
- $w^* \in M$.

Provided $M \subseteq \mathcal{H} \star w^*$, except with probability $2^{-\Omega(\lambda)}$, we have $\mathsf{CDH}(w, w') \in M_{w,w'}$, and so $\mathcal{H} \star w^* = \mathsf{CDH}(w^\star, \mathcal{H} \star x) \subseteq M$. Therefore, $M = \mathcal{H} \star w^*$ with probability at least $3/4 - 2^{-\Omega(\lambda)} \geq 2/3$. Since each iteration samples independently the distribution over M, by simple concentration bounds $\mathcal{H} \star w^\star$ will be the majority element of $\mathcal{M}$, except with probability $2^{-\Omega(\lambda)}$. □

Note that $\mathcal{A}_2^0$ runs $\mathcal{A}_1$ for $(T+s)|L|^2 = O(|L|^2\lambda/q^3) = \tilde{O}(q^{-5})$ times, giving $\tilde{O}(q^{-13})$ total queries to $\mathcal{A}$ and the group action operation. Meanwhile, $\mathcal{A}_2^1$ runs $\mathcal{A}_1$ for $\lambda|L|^2$ times, giving $\tilde{O}(q^{-10})$ queries to $\mathcal{A}$ and the group operation. From this point on, we fix a single $L \leftarrow \mathcal{A}_2^0()$ once and for all.

Removing Superfluous Information. We will next want to run quantum period-finding algorithms which make queries to $\mathcal{A}_2^1$ on superpositions of inputs. These algorithms, however, assume $\mathcal{A}_2^1$ 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 $\mathcal{A}_2^1$ and then un-computing all the intermediate values. The result is that $\mathcal{A}_2^1$ is negligibly close to behaving as the function mapping $(y,z) \mapsto \mathcal{H} \star \mathsf{CDH}(y,z)$. From now on, we will therefore assume that $\mathcal{A}_2^1$ is such a function.

Computing $\mathcal{H}$. Given algorithm $\mathcal{A}_2^1$, we can compute the subgroup $\mathcal{H}$ using quantum period-finding [BL95]. Concretely, the function $a \mapsto \mathcal{A}_2^1(a\star x, x, L)$ will output $(a\mathcal{H}) \star x$, which is periodic with set of periods $\mathcal{H}$. Therefore, applying quantum period finding to the procedure $a \mapsto \mathcal{A}_2^1(a\star x, x, L)$ will recover $\mathcal{H}$. This will make $O(\log|G|)$ calls to $\mathcal{A}_2^1(a \star x, x, L)$.

Solving DLog in $G/\mathcal{H}$. Notice that $\mathcal{A}_2^1$ is a (near) perfect CDH-solver, just in the group action corresponding to $G/\mathcal{H}$. Concretely, the group $G/\mathcal{H}$ acts on the set $X/\mathcal{H} := \{\mathcal{H} \star y : y \in X\}$ in the obvious way; the distinguished element of $X/\mathcal{H}$ is $\mathcal{H} \star x$. Our algorithm $\mathcal{A}_2^1$ gives a perfect CDH algorithm for this group action: we compute $\mathsf{CDH}(\mathcal{H}\star y, \mathcal{H}\star z)$ as $\mathcal{A}_2^1(y', z')$ for an arbitrary $y' \in \mathcal{H}\star y, z' \in \mathcal{H}\star z$.

We apply Galbraith et al. [GPSV18] to our CDH adversary for $(G/\mathcal{H}, X/\mathcal{H})$ to obtain a DLog adversary $\mathcal{B}(g\mathcal{H} \star x)$ which computes $g\mathcal{H}$. For completeness, we sketch the idea: Let $\mathbf{a}$ be a set of generators for $G/\mathcal{H}$. 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 $h \star (\mathcal{H} \star x)$, to compute $h^y \star (\mathcal{H} \star x)$ in $O(\log y)$ steps using repeated squaring. Given a DLog instance $g \star (\mathcal{H} \star x) = \mathbf{a}^{\mathbf{v}} \star (\mathcal{H} \star x)$, we define the function $(\mathbf{x}, y) \mapsto \mathbf{a}^{\mathbf{x}+y\mathbf{v}} \star (\mathcal{H} \star x)$, which can be computed using the CDH oracle. 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 $G/\mathcal{H}$. We now turn this into a full DLog adversary, which works as follows:

- Given $y = c \star x$, first apply the DLog adversary for $G/\mathcal{H}$, which outputs $c\mathcal{H}$.

– For each $a \in c\mathcal{H}$ (which is polynomial sized), test if $y = a \star x$. We output the unique such a.

Overall, assuming q is small relative to $\log |G|$, the running time of the algorithm is dominated by the cost of running $\mathcal{A}_2^0$, namely $\tilde{O}(q^{-13})$ total calls to $\mathcal{A}$ and the group action operations.

Remark 4. The dependence on q in our reduction is not ideal. The cost of our attack, however, is dominated by the cost of determining the subgroup. Typically, however, we expect the possible small-order subgroups to be known, and for there to only be a very limited number of options. In this case, we expect the complexity of our attack could be drastically improved.

3.1 Extending to Non-Regular Group Actions

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 L outputted by $\mathcal{A}_1$ may no longer be a subset of $\mathcal{H}_\delta \star x$, as it may have pieces from elements from different orbits. But $L \cap G \star x$ is still a subset of $\mathcal{H}_\delta \star x$, and similar statements hold for $\mathcal{A}_2^0, \mathcal{A}_2^1$ as well. This is enough to ensure that we obtain a near-perfect CDH algorithm on $(G/S)/\mathcal{H}$.

4 On the DDH and CDH (In)equivalence

A natural question to ask is whether we can show that the group action variants of CDH and DDH are equivalent. In traditional groups, there are a number of ways to argue that CDH and DDH are not equivalent, including by positing the existence of bilinear maps [BF01].

We show that for general group actions, the problems are also *not* equivalent. We do this by providing examples of group actions where "CDH" is hard and "DDH" is easy. In particular, we show that any group action where the group can be written as a non-trivial product group has the potential to be "CDH" hard but not "DDH" hard. This mirrors what we know classically and in the plain group setting, since there we can have groups that are CDH hard but not DDH hard. We state this formally in the following lemma.

Lemma 13. *Let $(G, X, \star)$ be an effective group action such that no efficient adversary can solve the group action CDH problem (as defined in definition 5) over it. Then there exists a group action $(G', X', \bigstar)$ where no efficient adversary can solve the CDH problem, but there exists a PPT algorithm for solving the group action DDH problem (as defined in definition 6).*

Proof. Consider some extra group $\tilde{G}$. We can define a "group action" $\tilde{G} \times \tilde{G} \to \tilde{G}$ where the group action operation is simply group multiplication in $\tilde{G}$. Discrete log is trivial on this group since group inversion is efficient.

From our secure group action $(G, X, \star)$ and our insecure "group action," we construct another group action $(G', X', \bigstar)$ which we define as follows:

$$G' = G \times \tilde{G}$$

$$X' = X \times \tilde{G}$$

$$\bigstar : \left\{G \times \tilde{G}\right\} \times \left\{X \times \tilde{G}\right\} \to \left\{X \times \tilde{G}\right\}$$

For some $g \in G$, $x \in X$, $\tilde{g}_1, \tilde{g}_2 \in \tilde{G}$, we define the action as follows:

$$\{g, \tilde{g}_1\} \bigstar \{x, \tilde{g}_2\} = \{g \star x, \tilde{g}_1 \tilde{g}_2\}$$

Note that this definition meets all of the requirements of the group action. $G \times \tilde{G}$ is a (product) group, and all of the group action axioms hold.

We can immediately build a PPT distinguisher: given a DDH tuple $(x_1' = (x, \tilde{g}_1), g' \star x_1' = (g \star x_1, \tilde{g}\tilde{g}_1), x_2' = (x, \tilde{g}_2), g' \star x_2' = (g \star x_2, \tilde{g}\tilde{g}_2))$, we can perform the following check:

$$(\tilde{g}\tilde{g}_2)^{-1} (\tilde{g}\tilde{g}_1) = \tilde{g}_2^{-1}\tilde{g}_1$$

This immediately breaks the pseudorandomness of the group action, meaning that the group action DDH problem is not hard over $\left(\tilde{G}, \tilde{X}, \bigstar\right)$. However, any adversary that breaks the group action CDH problem on $\left(\tilde{G}, \tilde{X}, \bigstar\right)$ also breaks it on $(G, X, \star)$, which contradicts our assumption that the CDH problem is hard on this group action. $\square$

In the above example, we used a product group. A nice question is as follows: what happens if we assume that the group must be, say, prime-order cyclic? This case is much harder to show interesting results since we don't have efficiently computable bilinear pairings as in the standard group setting.

5 A Generic Group Action Framework

In this section, we define a generic group action framework. We create two models: one for classical queries, and one which allows quantum queries. Our framework is based on the generic group framework of Shoup [Sho97]. We borrow from Shoup's description in our own explanation below.

Let G be a group of order n, let X be a set that is representable by bit strings of length m, and let $(G, X, \star)$ be a group action. We define additional sets S_G and S_X such that they have cardinality of at least n and 2^m, respectively. We define *encoding functions* of σ_G and σ_X on S_G and S_X, respectively, to be injective maps of the form $\sigma_G : G \to S_G$ and $\sigma_X : X \to S_X$.

A generic algorithm $\mathcal{A}$ for $(G, X, \star)$ on (S_G, S_X) is a probabilistic algorithm that behaves in the following way. It takes as input two *encoding lists* $(\sigma_G(g_1), ..., \sigma_G(g_k))$ and $(\sigma_X(x_1), ..., \sigma_X(x_{k'}))$ where each $g_i \in G$ and $x_i \in X$ and where σ_G and σ_X are encoding functions of G on S_G and X on S_X, respectively. As the algorithm executes, it may consult two oracles, $\mathcal{O}_G$ and $\mathcal{O}_X$.

The oracle $\mathcal{O}_G$ takes as input two strings y, z representing group elements and a sign "+" or "–", computes $\sigma_G\left(\sigma_G^{-1}(y) \pm \sigma_G^{-1}(z)\right)$. The oracle $\mathcal{O}_X$ takes as input a string y representing a group element and string z representing a set element, and computes $\sigma_X\left(\sigma_G^{-1}(y) \star \sigma_X^{-1}(z)\right)$. As is typical in the literature, we can force all queries to be on either the initial encoding lists or the results of previous queries by making the string length m very long. We typically measure the running time of the algorithm by the number of oracle queries.

We can also extend the generic group action model to the quantum setting, where we allow *quantum* queries to the oracles. We model quantum queries in the usual way: $\mathcal{O}_G \sum_{y,z,\pm,w} \alpha_{y,z,\pm,w} |y, z, \pm, w\rangle = \sum_{y,z,\pm,w} \alpha_{y,z,\pm,w} |y, z, \pm, w \oplus \mathcal{O}_G(y, z, \pm)\rangle$ and $\mathcal{O}_X \sum_{y,z,w} \alpha_{y,z,w} |y, z, w\rangle = \sum_{y,z,w} \alpha_{y,z,w} |y, z, w \oplus \mathcal{O}_X(y, z)\rangle$.

6 On REGAs

Our reductions showing the equivalence of group action DLog and CDH unfortunately only hold for EGAs and not for REGAs. In their work showing an equivalence for a perfect oracle [GPSV18], Galbraith *et al.* suggest that applying the BKZ algorithm [SE94] or other lattice reduction techniques can be used to complete the reduction. In this section, we formalize this idea with a number of resuts on the relationship between REGAs and lattices, and, in particular, focus on the 1D-SIS problem, which is a lattice problem that is equivalent to the standard form of LWE modulo PPT reductions. Due to space constraints, we only state the relevant lemmas in this section and defer proofs to the full version of the paper. We present the full, unabridged version of this section as well as the formal definitions related to REGAs in full in the full version of the paper.

In this section, we will rely on the fact that, using a generalization of Shor's algorithm [CM01], we can (quantumly) efficiently compute the isomorphism between any abelian group G and a product group over groups of the integers

$$G \cong \mathbb{Z}_1 \times ... \times \mathbb{Z}_m.$$

We additionally note that most of our results here only hold for *regular* group actions. We do not consider this a major drawback since all popular REGAs (e.g. CSIDH and its derivatives) are regular REGAs.

A "1D-SIS Oracle" Completes the DLog/CDH Reduction for REGAs. We begin by formalizing the argument from Galbraith *et al.* [GPSV18] that efficient lattice reductions could be used to show the discrete log/CDH equivalence of REGAs. While doing this in full would involve completely replicating our earlier proof, we simply point out at which stages using a REGA makes a difference and how we can handle these points.

We first need to ensure that we can randomly sample elements from a REGA. We define a notion of "sampleable REGA" capturing this:

Definition 7. ***Sampleable REGA****: Let $(G, X, \star)$ be a REGA with group element vector $\mathbf{g} = (\mathbf{g}_1, ..., \mathbf{g}_m)$ for some m. We say that such a REGA is sampleable if there exists an efficient way to sample a vector $\mathbf{b} \in \{-\gamma, \gamma\}^m$ for some polynomial γ such that the vector $\mathbf{r} = \sum_{i=1}^m \mathbf{b}_i \mathbf{g}_i$ is distributed statistically close to uniform over G.*

This requirement essentially just requires that some form of the leftover hash lemma applies over the group with the action-computable elements as the "base." We note that many cryptosystems build on REGAs (i.e. those using CSIDH) implicitly make this assumption. We need this to rule out cases where the elements of $\mathbf{g}$ are too clustered: for instance, if G is Z_p and all of the $\mathbf{g}_i$ are small integers, we will not be able to effectively compute the group action on randomly distributed group elements. Next, we define a specialized problem we call "REGA-SIS." Note that this is not a standard problem because, among other things, the $\mathbf{g}_i$ distribution comes from the definition of the REGA.

Definition 8. ***REGA-SIS****: Let $(G, X, \star)$ be a REGA with group element vector $\mathbf{g} = (\mathbf{g}_1, ..., \mathbf{g}_m)$ for some m. We define $SIS_{REGA,\beta}$ in the following way: given a random element $h \leftarrow G$, the problem is to find some vector $\mathbf{u} \in [-\beta, \beta]^m$ such that $h = \sum_{i=1}^m \mathbf{u}_i \mathbf{g}_i$.*

This problem is parameterized by the REGA and, in particular, by both the group and the computable elements. Furthermore, for $G = \mathsf{Z}_q$ and when each coefficient of $\mathbf{g}$ is distributed uniformly at random, REGA-SIS is exactly the 1D-inhomogeneous SIS (1D-ISIS) problem (which is reducible to standard 1D-SIS with a slight loss in parameters, and 1D-SIS itself is again reducible to and from standard LWE, for appropriate parameter settings). So this problem can be viewed as a slightly unnatural generalization of SIS. We can now state our core lemma on REGAs.

Lemma 14. *Consider any efficiently sampleable REGA as defined in definition 7. Then any adversary that can solve the CDH problem on the REGA with advantage ϵ_1 and the $SIS_{REGA,\beta}$ problem for the same REGA and some polynomial β with advantage ϵ_2 can be used to solve the discrete log problem on the same REGA with advantage $\epsilon_1\epsilon_2$.*

Discrete Log on REGAs and 1D-SIS. Recall that a REGA is a group action $(G, X, \star)$ where the action is only computable on a set of group elements defined by a vector $\mathbf{g} = (g_1, \ldots, g_n)$. Suppose that G is an abelian group. We claim that if these group elements are distributed randomly, then any adversary that can solve discrete log on the REGA can be used to solve the 1D-SIS problem for certain parameter settings (which are all reducible to some form of standard LWE). The analysis of most practical REGAs (e.g. CSIDH) assume follow this convention, so this is not an unreasonable assumption to make. We formalize this with the following lemma.

Lemma 15. *Let q and m be integers such that $m \geq 3 \log q$. Let $\mathcal{A}$ be an adversary that can solve the group action DLog problem on regular REGAs of the form $(Z_q, X, \star)$ where the vector of group elements $\mathbf{g} = (g_1, ..., g_m)$ is m elements long and distributed uniformly at random with advantage ϵ. Then $\mathcal{A}$ can be used to solve the $1D\text{-}SIS_{m,q,\beta}$ problem for some polynomial β with advantage ϵ.*

CDH on REGAs. We above showed that an adversary that can solve discrete log on a REGA can solve a variant of the SIS problem, and that any adversary that can solve this SIS variant can also be used to complete the CDH/DLog reduction. Can we tie all of this together to get an unconditional CDH to DLog reduction to work for REGAs?

We give some mild evidence in this direction. We can show that any *generic* adversary that makes only *classical* queries to a generic group action oracle (that may still be able to perform quantum computations) can be used to solve the REGA-SIS problem we defined above in Definition 8. We can then use this to complete the CDH to DLog reduction for generic, classically-querying adversaries. Of course, classically we can prove CDH and DLog are unconditionally hard (this follows from the unconditional hardness of these problems in plain groups), and therefore equivalent. But phrasing the equivalence as a reduction suggests a possible starting point for a quantum equivalence

Lemma 16. *Consider some regular, abelian, and efficiently sampleable REGA $(G, X, \star)$ with computable elements $\mathbf{g} = (\mathbf{g}_1, ..., \mathbf{g}_m)$. Suppose there exists a* generic *adversary* making only classical group and group action queries *that can solve the GA-CDH problem on this REGA with advantage ϵ. Then there exists an adversary that can solve REGA-SIS for some polynomial parameter β with advantage $\epsilon/2$.*

Discussion. We have shown three core results on REGAs (stated informally): an adversary for our REGA-SIS problem would complete our CDH/DLog reduction for REGAs, an adversary for DLog on REGAs solves this REGA-SIS problem, and a generic adversary that only makes classical queries that can solve CDH on REGAs can be used to solve REGA-SIS as well. All together, these seemingly tightly bind CDH and DLog on a REGA to a SIS-like problem that appears to be vulnerable to lattice-based cryptanalysis [GPSV18]. We therefore provide some evidence for a quantum DLog-CDH equivalence on REGAs.

7 Hidden Subgroup Problems and GAs

In this section, we discuss some similarities between different kinds of hidden subgroup problems (HSPs) and solving group actions. We particularly focus on the *generalized* dihedral group. We note that, among other things, formalizing a connection between group actions and these kinds of problem would allow us to potentially tie two of the most popular forms of post-quantum cryptosystems (lattices and isogenies) together. Once again, due to space constraints, we just state lemmas here and defer the full presentation to the full version of the paper.

The Generalized Dihedral Hidden Subgroup Problem. We begin by defining the *generalized* dihedral group.

Definition 9. ***Generalized Dihedral Group****: Let A be an abelian group. The* generalized dihedral group *on A, denoted D_A, is the group defined by $\mathbb{Z}_2 \ltimes A$.*

When $A \cong \mathbb{Z}_n$, we get back the standard notion of the dihedral group on $2n$ elements. The dihedral group has a number of nice geometric explanations and properties, but we defer those to others [KLG06]. We next define the general dihedral hidden subgroup problem. However, rather than defining this problem in its traditional sense, we will use an equivalent formulation known as the *abelian hidden shift problem.* These problems are well known to be equivalent [CVD05].

Definition 10. ***Abelian Hidden Shift Problem (equivalent to GDHSP)****: Consider some functions f, g such that, for some $c \in A$ and for all $b \in \mathbb{Z}_n$, $f(b) = g(b + c)$. We also require that each of the $||A||$ output values of f and g are also distinct. We say that an algorithm solves the abelian hidden shift problem if, given descriptions of f and g, it outputs c (which reveals the subgroup in the generalized dihedral hidden subgroup version of the problem).*

The dihedral hidden subgroup problem has strong connections to lattice problems [Reg02], in that if an efficient algorithm for the DHS problem that uses a special type of "coset sampling" exists, then an efficient algorithm for the LWE problem exists as well. The best known algorithms for solving the DHS problem are subexponential and based on Kuperberg's algorithm [Kup05, Reg04, Kup13].

An Algorithm for the AHSP Breaks Regular, Abelian Group Actions. We first show a relatively straightforward result: any algorithm that can solve the abelian hidden shift problem can be used to solve DLog on a regular, abelian group action. This is essentially already folklore since there have been many instances (starting with [CJS14]) using Kuperberg's algorithm or related principles to build attacks against isogenies that can be modelled as EGAs.

Lemma 17. *Let $(G, X, \star)$ denote a regular, abelian group action. Suppose there exists a PPT algorithm $\mathcal{A}$ for solving the abelian hidden shift problem on A with probability ϵ. Then there exists for solving the GA-DLog problem on $(G, X, \star)$ with probability ϵ.*

Using Group Action Algorithms to Solve the AHSP. What about the other direction? Can we show that an adversary that can break DLog on a group action can solve the AHSP? Unfortunately, this seems difficult: because the AHSP is described so generally–the functions f and g can be anything as long as the functions are injective–so it seems difficult or impossible to prove this for any non-generic algorithm.

But what about generic algorithms? Could we prove that the AHSP is equivalent to generically solving DLog over group actions? This seems like it might be plausible. The most interesting result would show equivalence in a generic group action model with quantum queries. While this may be attainable, unfortunately we do not know how to achieve this result. However, we can show that an adversary that can generically solve group action DLog with classical queries can be used to solve the AHSP, which is seemingly a step in the right direction. We formalize this result below.

Lemma 18. *Let $(G, X, \star)$ be an abelian, regular group action (EGA). Suppose there exists a generic adversary $\mathcal{A}$ that breaks the group action DLog problem (as defined in definition 4) with advantage ϵ on this group action. Then there exists an algorithm that solves that AHSP on G with advantage ϵ.*

Discussion. Unfortunately, it seems difficult to show a full quantum equivalence between the generalized dihedral hidden subgroup problem and solving DLog on a generic group action. The challenge comes from the fact that it is difficult quantumly to "remember" an adversary's query for later use in the simulation. One possible direction is to use compressed oracles [Zha19], which offer some ability to record quantum queries. However, it appears challenging to adapt the compressed oracle framework to highly structured oracles such as generic group actions. Nevertheless, we close this section with the following conjecture, which we think is very interesting future work:

Conjecture 1. The generalized dihedral hidden subgroup problem on an abelian group A is equivalent to the group action discrete logarithm problem on a regular, abelian group action $(A, X, \star)$ in a quantum generic model.

References

ADMP20. Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *ASIACRYPT 2020, Part II*, LNCS, pages 411–439. Springer, Heidelberg, December 2020.

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BGLS19. Shi Bai, Steven D. Galbraith, Liangze Li, and Daniel Sheffield. Improved combinatorial algorithms for the inhomogeneous short integer solution problem. *Journal of Cryptology*, 32(1):35–83, January 2019.

BKM17. Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable PRFs from standard lattice assumptions. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 415–445. Springer, Heidelberg, April / May 2017.

BKT14. Adi Ben-Zvi, Arkadius Kalka, and Boaz Tsaban. Cryptanalysis via algebraic spans. Cryptology ePrint Archive, Report 2014/041, 2014. https://eprint.iacr.org/2014/041.

BKT18. Adi Ben-Zvi, Arkadius G. Kalka, and Boaz Tsaban. Cryptanalysis via algebraic spans. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 255–274. Springer, Heidelberg, August 2018.

BKV19. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *ASIACRYPT 2019, Part I*, LNCS, pages 227–247. Springer, Heidelberg, December 2019.

BL95. Dan Boneh and Richard J. Lipton. Quantum cryptanalysis of hidden linear functions (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 424–437. Springer, Heidelberg, August 1995.

BL96. Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 283–297. Springer, Heidelberg, August 1996.

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

BMZ19. James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Heidelberg, August 2019.

BS20. Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, LNCS, pages 493–522. Springer, Heidelberg, May 2020.

BV15. Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015.

BY91. Gilles Brassard and Moti Yung. One-way group actions. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 94–107. Springer, Heidelberg, August 1991.

CD22. Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975.

CJS14. Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.

CK20. Leonardo Colò and David Kohel. Orienting supersingular isogeny graphs. *Journal of Mathematical Cryptology*, 14(1):414–437, 2020.

CLM+18. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018.

CM01. Kevin K. H. Cheung and Michele Mosca. Decomposing finite abelian groups. *Quantum Information & Computation*, 1(3):26–32, 2001.

Cou06. Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. https://eprint.iacr.org/2006/291.

CVD05. Andrew M Childs and Wim Van Dam. Quantum algorithm for a generalized hidden shift problem. *arXiv preprint quant-ph/0507190*, 2005.

DDF21. Pierrick Dartois and Luca De Feo. On the security of osidh. *Cryptology ePrint Archive*, 2021.

den90. Bert den Boer. Diffie-Hellman is as strong as discrete log for certain primes (rump session). In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 530–539. Springer, Heidelberg, August 1990.

DG19. Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, Heidelberg, May 2019.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DJP14. Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.

DM20a. Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In *PKC 2020, Part II*, LNCS, pages 187–212. Springer, Heidelberg, 2020.

DM20b. 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*, pages 187–212, Cham, 2020. Springer International Publishing.

GPSV18. Steven Galbraith, Lorenz Panny, Benjamin Smith, and Frederik Vercauteren. Quantum equivalence of the DLP and CDHP for group actions. Cryptology ePrint Archive, Report 2018/1199, 2018. https://eprint.iacr.org/2018/1199.

ILL89. Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989.

JQSY19. Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun. General linear group action on tensors: A candidate for post-quantum cryptography. In *TCC 2019, Part I*, LNCS, pages 251–281. Springer, Heidelberg, March 2019.

KLC+00. Ki Hyoung Ko, Sangjin Lee, Jung Hee Cheon, Jae Woo Han, Ju-Sung Kang, and Choonsik Park. New public-key cryptosystem using Braid groups. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 166–183. Springer, Heidelberg, August 2000.

KLG06. Hirotada Kobayashi and François Le Gall. Dihedral hidden subgroup problem: A survey. *Information and Media technologies*, 1(1):178–185, 2006.

Kup05. Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal of Computing*, 35(1):170–188, 2005.

Kup13. Greg Kuperberg. Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. In Simone Severini and Fernando Brandao, editors, *8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013)*, volume 22 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20–34, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

LGdSG21. Yi-Fu Lai, Steven D Galbraith, and Cyprien Delpech de Saint Guilhem. Compact, efficient and uc-secure isogeny-based oblivious transfer. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 213–241. Springer, 2021.

Mas90. P. Massart. The Tight Constant in the Dvoretzky-Kiefer-Wolfowitz Inequality. *The Annals of Probability*, 18(3):1269 – 1283, 1990.

Mau94. Ueli M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 271–281. Springer, Heidelberg, August 1994.

MM22. Luciano Maino and Chloe Martindale. An attack on sidh with arbitrary starting curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. https://eprint.iacr.org/2022/1026.

MW96. Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 268–282. Springer, Heidelberg, August 1996.

Onu21. Hiroshi Onuki. On oriented supersingular elliptic curves. *Finite Fields and Their Applications*, 69:101777, 2021.

Pei20. Chris Peikert. He gives C-sieves on the CSIDH. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, LNCS, pages 463–492. Springer, Heidelberg, May 2020.

Reg02. Oded Regev. Quantum computation and lattice problems. In *43rd FOCS*, pages 520–529. IEEE Computer Society Press, November 2002.

Reg04. Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv:quant-ph/0406151, June 2004.

Rob22. Damien Robert. Breaking sidh in polynomial time. Cryptology ePrint Archive, Paper 2022/1038, 2022. https://eprint.iacr.org/2022/1038.

RS06. Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. https://eprint.iacr.org/2006/145.

SE94. Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1):181–199, 1994.

Sho94. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

Shp08. Vladimir Shpilrain. Cryptanalysis of stickel's key exchange scheme. In *International Computer Science Symposium in Russia*, pages 283–288. Springer, 2008.

Sti05. Eberhard Stickel. A new method for exchanging secret keys. In *Third International Conference on Information Technology and Applications (ICITA'05)*, volume 2, pages 426–430. IEEE, 2005.

SU05a. Vladimir Shpilrain and Alexander Ushakov. A new key exchange protocol based on the decomposition problem. Cryptology ePrint Archive, Report 2005/447, 2005. https://ia.cr/2005/447.

SU05b. Vladimir Shpilrain and Alexander Ushakov. Thompson's group and public key cryptography. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 151–163. Springer, Heidelberg, June 2005.

Zha19. Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, Heidelberg, August 2019.