Generic Models for Group Actions

Julien Duman, Dominik Hartmann, Eike Kiltz,
Sabrina Kunzweiler, Jonas Lehmann, Doreen Riepel

Ruhr-Universität Bochum, Germany
`{julien.duman, dominik.hartmann, eike.kiltz, sabrina.kunzweiler, jonas.lehmann-c6j, doreen.riepel}@rub.de`

Abstract. We define the Generic Group Action Model (GGAM), an adaptation of the Generic Group Model to the setting of group actions (such as CSIDH). Compared to a previously proposed definition by Montgomery and Zhandry (ASIACRYPT '22), our GGAM more accurately abstracts the security properties of group actions.

We are able to prove information-theoretic lower bounds in the GGAM for the discrete logarithm assumption, as well as for non-standard assumptions recently introduced in the setting of threshold and identification schemes on group actions. Unfortunately, in a natural quantum version of the GGAM, the discrete logarithm assumption does not hold.

To this end we also introduce the weaker Quantum Algebraic Group Action Model (QAGAM), where every set element (in superposition) output by an adversary is required to have an explicit representation relative to known elements. In contrast to the Quantum Generic Group Action Model, in the QAGAM we are able to analyze the hardness of group action assumptions: We prove (among other things) the equivalence between the discrete logarithm assumption and non-standard assumptions recently introduced in the setting of QROM security for Password-Authenticated Key Exchange, Non-Interactive Key Exchange, and Public-Key Encryption.

Keywords: Group Actions, CSIDH, Algebraic Group Action Model, Generic Group Action Model

1 Introduction

Group Actions. Group actions are considered a promising candidate for building post-quantum secure cryptography. While similar to the well-known prime-order groups, group actions are more limited and provide less structure. For a group $(\mathcal{G}, \circ)$ with neutral element $e \in \mathcal{G}$ and a set $\mathcal{X}$, a group action is a map

$$\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$$

that is compatible with the group operation in $\mathcal{G}$. That is, $e \star x = x$ for all $x \in \mathcal{X}$ and $(g \circ h) \star x = g \star (h \star x)$ for all $g, h \in \mathcal{G}$ and $x \in \mathcal{X}$. It can be thought of as an analogue to the exponentiation in a multiplicative prime-order group, which leads to natural analogues of problems such as the discrete logarithm problem in group actions (GA-DLOG). The similarity to prime-order groups allows the adaptation of several basic schemes to group actions [18,39,15,43,30,20]. Crucially, there is no group law on $\mathcal{X}$. This makes group actions resilient against well-known quantum attacks on groups such as Shor's algorithm [41]. To date, the best known quantum attacks on group actions are based on Kuperberg's algorithm [31] which has subexponential runtime.

The most prominent example for a cryptographic group action to date is the CSIDH group action [15], which is based on isogenies between supersingular elliptic curves. While promising, isogeny-based cryptography is still fairly new and has not been as thoroughly studied as traditional cryptography based on the discrete logarithm problem in prime order groups or the RSA problem. This lack of analysis was exemplified by the recent attacks on the SIDH assumption [14,32,38], which completely break SIDH and related schemes, but have no impact on the security of CSIDH.

Generic Models. A useful tool for analyzing cryptographic problems are generic models. One popular model in the setting of prime-order groups is Shoup's Generic Group Model (GGM) [42]. The GGM replaces group elements with random labels (without any algebraic meaning) and only allows computation of the group law via an oracle. In the GGM one can provide information-theoretic lower bounds on the number of calls to the group oracle for certain cryptographic problems. For example, in the GGM the discrete logarithm problem over groups of prime order p can only be solved with at least $\sqrt{p}$ calls to the group oracle.

Another useful model in the setting of prime-order groups is the Algebraic Group Model (AGM) [23]. In the AGM all algorithms know the group structure and can compute group operations. However, adversaries are restricted to only producing new group elements by combining previously known group elements via the group law, i.e. they have to behave "algebraically". This is enforced by requiring algebraic adversaries to provide a representation of their output group elements relative to their inputs. While this model cannot be used to prove lower bounds for cryptographic assumptions, it has proven useful for relating the hardness of different assumptions and cryptographic protocols [23,34,8].

Generic Models for Group Actions. The GGM has recently been adapted to the group action setting by Montgomery and Zhandry [35] in order to prove the generic quantum equivalence of GA-DLOG and GA-CDH in the setting of restricted effective group actions (REGA).[1] We refer to this as the MZ-Generic Group Action Model (MZ-GGAM). The MZ-GGAM encodes elements from the group $\mathcal{G}$ *and* the set $\mathcal{X}$ with random labels and provides oracles for both the group law in $\mathcal{G}$ and the group action $\star$ on $\mathcal{X}$. Additionally, they define the MZ-Quantum Generic Group Action Model (MZ-QGGAM), a quantum analogue of the MZ-GGAM providing quantum access to the two oracles.

While the definition of the MZ-GGAM seems reasonable when considering REGAs, we believe that it does not accurately capture the security properties of general effective group actions (EGA). Why? One could simply use group $\mathcal{G}$ and ignore set $\mathcal{X}$ and mapping $\star$. In the MZ-GGAM one can prove the hardness of standard assumption like the discrete logarithm in group $\mathcal{G}$, even though group actions are "not supposed" to source their hardness from problems over $\mathcal{G}$. This simple observation allows to provably port all standard group-based cryptography to the MZ-GGAM. That is, in the MZ-GGAM group actions actually have more structure than prime-order groups (though they should not). Furthermore, in the MZ-Quantum Generic Group Action Model one can efficiently compute discrete logarithms over $\mathcal{G}$ by applying Shor's algorithm [41], hence learn the entire group structure of $\mathcal{G}$. This shows that hiding the structure of group $\mathcal{G}$ with random labels is useless in a quantum setting.

1.1 Contributions

We propose an alternative definition of the generic group model for group actions (GGAM) and analyze several standard and non-standard assumptions in it. We also consider a quantum version, QGGAM, and show that the group action discrete logarithm problem (GA-DLOG) does *not* hold in it. Furthermore, we define a (quantum) algebraic group model for group actions, AGAM and QAGAM, and use it to relate the hardness of several useful group-action assumptions.

We will now go over our results in a bit more detail.

Generic Group Action Model. We propose a new definition of the Generic Group Action Model (GGAM), which differs from the previous definition MZ-GGAM of [35] in that we only encode elements from the set $\mathcal{X}$ and not from group $\mathcal{G}$. Our definition assumes $\mathcal{G}$ to be cyclic of order N (see Appendix A for a generalization to non-cyclic abelian group actions). In a quantum setting the isomorphism between $\mathcal{G}$ and $\mathbb{Z}_N$ is efficiently computable with Shor's algorithm, so we can assume it to be known with some quantum precomputation. Hence in our GGAM we set $\mathcal{G} := \mathbb{Z}_N$, which also models the fact that a group action should not source its hardness from group $\mathcal{G}$, an important security feature of group actions.

Relation to the Generic Group Model. While it seems intuitive that hardness results in the GGM (over prime-order groups) should carry over to the more restricted GGAM (over group actions), there are some subtleties in formalizing this. We prove a "lifting lemma" which states that all hardness results in the GGM for a group of prime order p carry over to the GGAM if the order N of the group action satisfies $N = p - 1$. Looking ahead, this restriction is a consequence of how the GGAM can be embedded in the GGM.

Generic Group Action Model with Twists. We extend the definition of the GGAM to the GGAM with Twists (GGAM$^\top$), which models group actions like CSIDH more closely. More specifically, we include

[1] In a REGA (not considered in this work) the group action evaluation cannot be performed efficiently for arbitrary group elements, but it is necessary to find a suitable representation of the element first. In particular, this is the case when the group structure is unknown.

a twisting algorithm that allows to compute $-\mathfrak{a} \star x$ from $\mathfrak{a} \star x$ efficiently for any $\mathfrak{a} \in \mathbb{Z}_N$ and $x \in \mathcal{X}$. This allows us to capture a wider variety of practical group action instantiations. Unfortunately, there is no analogue to twisting in the prime order group setting, therefore we can not prove a lifting lemma from the GGM to the GGAM$^\top$.

Generic Lower Bounds. Next, we prove explicit lower bounds on the success probability of generic adversaries on GA-DLOG, as well as two non-standard assumptions in the GGAM$^\top$ that were recently introduced in the context of threshold schemes [19], identification schemes [6] and password-authenticated key exchange [1]. Our proofs are rather straightforward and adapt well-known proof techniques from the GGM. The resulting bounds are almost the same bounds as in the GGM except for constant factors due to the (potentially) composite order of the group action. This actually highlights that there are cases where group action based assumptions become potentially easier. Our analysis makes it simple to detect and to subsequently exclude those cases in the assumption.

Generic Models in the Quantum Setting. We define the Quantum Generic Group Action Model (QGGAM), an extension of the GGAM to the quantum setting where the adversary has quantum access to the group action oracle. Similar to the classical setting, the group $\mathcal{G}$ is simply modeled as $\mathbb{Z}_N$. We first show that our QGGAM is in fact equivalent to the MZ-GGAM, while it is conceptually much simpler. Unfortunately, we observe that even GA-DLOG is information-theoretically *easy* in the QGGAM. This is due to a generic algorithm by Ettinger and Høyer [22], which breaks GA-DLOG with polynomially many quantum group-action queries plus (classical) exponential time to solve the Trigonometric Approximation Problem (TAP). TAP is a purely combinatorial problem which is independent of the group action. Since the QGGAM is an information-theoretic model which only counts the number of oracle queries, this constitutes an efficient quantum attack against GA-DLOG in the QGGAM. One interpretation of this result is that the quantum hardness of GA-DLOG is a combinatorial property rather than an algebraic one.

Algebraic Group Action Model. We define the group action analogue of the AGM [23] in the quantum setting, which we call the Quantum Algebraic Group Action Model with twists (QAGAM$^\top$) and without twists (QAGAM). While there cannot exist any meaningful bounds in the QGGAM, we are still able to quantum-relate the following assumptions in the QAGAM/QAGAM$^\top$.

- **Assumptions for KEM/NIKE.** The Group Action Quantum Strong Computational Diffie-Hellman assumption (GA-QSt-CDH) is the CDH assumption for group actions, where the adversary is furthermore given *quantum access* to (fixed-base) DDH oracles. GA-QSt-CDH assumptions are required to prove active security of group-action Hashed ElGamal encryption and Diffie-Hellman NIKE protocols in the QROM (used, for example, in Post-Quantum WireGuard and OPTLS) [20]. There is no known security analysis of GA-QSt-CDH. We prove that GA-QSt-CDH is equivalent to the GA-DLOG assumption in the QAGAM$^\top$, therefore giving the first indication towards its quantum security. Our proof relies on the semi-classical one-way to hiding lemma.
- **Assumptions for PAKE.** We consider the Group Action Quantum Strong Square-Inverse Diffie-Hellman (GA-QSt-SqInvDH) assumption [1] in the QAGAM$^\top$. This non-standard assumption is a combination of the Square-DH and Inversion-DH assumption relative to a flexible base, where an adversary is furthermore given quantum access to DDH oracles as in the GA-QSt-CDH assumption. The GA-QSt-SqInvDH assumption is required to prove security of a recently proposed PAKE protocol in the QROM [1]. We again prove that GA-QSt-SqInvDH is equivalent to the GA-DLOG assumption in the QAGAM$^\top$, therefore giving the first indication towards its quantum security.
- **Assumptions for ElGamal.** We study the Quantum CCA1 (QCCA1) security of the Group Action plain (unhashed) ElGamal KEM. QCCA1 security means that the adversary is allowed to ask the decryption oracle with ciphertexts in superposition, but only before seeing the challenge ciphertext. We prove its QCCA1 security to be equivalent to the Group Action q-Decisional Diffie-Hellman Problem (GA-q-DDH) in the QAGAM$^\top$.

2 Preliminaries

In this section, we fix some notation that will be used throughout the paper and recall standard definitions.

2.1 Notation

For integers m, n where $m < n$, $[m, n]$ denotes the set $\{m, m+1, ..., n\}$. For $m = 1$, we simply write $[n]$. By $\log(x)$ we denote the logarithm over the reals with base 2. For a (finite) set S, $s \xleftarrow{\$} S$ denotes that s is sampled uniformly and independently at random from S. $y \leftarrow \mathcal{A}(x_1, x_2, ...)$ denotes that on input $x_1, x_2, ...$ the probabilistic algorithm $\mathcal{A}$ returns y. $\mathcal{A}^{\mathrm{O}}$ denotes that algorithm $\mathcal{A}$ has access to oracle O. An adversary is a probabilistic algorithm. The notation $[\![B]\!]$, where B is a boolean statement, refers to a bit that is 1 if the statement is true and 0 otherwise.

2.2 Security Games

We define code based security games similar to [9]. A game G is defined as an algorithm (which we call the challenger) that provides a main procedure and (possibly zero) oracle procedures. An algorithm $\mathcal{A}$ playing game G gets an initial input from the challenger and can subsequently interact with (possibly zero) oracles. In the case of quantum algorithms, the oracles can be queried in superposition. At the end of its execution, $\mathcal{A}$ has to provide some output to the challenger. We say that $\mathcal{A}$ *wins* (or *solves*) the game G if the challenger accepts the output, which we write as $\mathrm{G}^{\mathcal{A}} \Rightarrow 1$. We define the *success probability* of $\mathcal{A}$ as $\Pr[\mathrm{G}^{\mathcal{A}} \Rightarrow 1]$. Note that we will use the words "game" and "problem" interchangeably.

2.3 Quantum Preliminaries

We recall some quantum computation preliminaries and notation as stated in [21].

QUBIT. A qubit $|x\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ is a 2-dimensional unit vector with coefficients in $\mathbb{C}$, i.e. $x = (\alpha, \beta) \in \mathbb{C}^2$ fulfilling the normalization constraint $|\alpha|^2 + |\beta|^2 = 1$. When neither $\alpha = 1$ nor $\beta = 1$, we say that $|x\rangle$ is in *superposition*.

n-QUBIT STATE. An n-bit quantum register $|x\rangle = \sum_{i=1}^{2^n-1} \alpha_i\,|i\rangle$ is a unit vector of $\mathbb{C}^{2^n} = (\mathbb{C}^2)^{\otimes n}$, that is $\alpha_i \in \mathbb{C}$ and $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$. We call the set $\{|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle\}$ the *computational basis*. When $|x\rangle$ can not be written as the tensor product of single qubits, we say that $|x\rangle$ is *entangled*.

MEASUREMENT. Unless otherwise stated, measurements are done in the computational basis. After measuring a quantum register $|x\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle$ in the computational basis, the state *collapses* and $|x\rangle = \pm\,|i\rangle$ with probability $|\alpha_i|^2$.

QUANTUM ALGORITHMS. A quantum algorithm $\mathcal{A}$ is a series of unitary operations U_i, where unitary operations are defined as to map unit vectors to unit vectors, preserving the normalization constraint of quantum registers. A quantum oracle algorithm $\mathcal{A}^{\mathrm{O}}$ is defined similarly, except it can query the oracle O after (or before) executing a unitary U_i. Since quantum computation needs to be reversible, we model an oracle $\mathrm{O} : X \to Y$ by a unitary U_{O} that maps $|x\rangle\,|y\rangle \mapsto |x\rangle\,|y \oplus \mathrm{O}(x)\rangle$. We only consider sequential quantum algorithms.

QUANTUM-ACCESS OF ORACLES. For an oracle O, we are going to write $|\mathrm{O}\rangle$ to denote that it can be queried on quantum inputs and O if it can not (which means that its inputs are implicitly measured). For an oracle which allows partial quantum-access, we write $|\cdot\rangle$ to denote the inputs which are quantum (i.e., not measured), for example $\mathrm{O}(\cdot, |\cdot\rangle)$ means that the first input is classical (i.e., implicitly measured on query) and the second is quantum. Alternatively to $|\mathrm{O}\rangle$ we might also write $\mathrm{O}(|\cdot\rangle, |\cdot\rangle)$, if O takes two inputs. While such a (partially) quantum oracle formally needs an additional ancillary input to write its result, we omit this to keep the interfaces between classical and quantum oracles aligned.

ONEWAY-TO-HIDING LEMMA. We recall the oneway-to-hiding lemma, which is used to reprogram (random) oracle values. Informally, the lemma states that if an oracle is reprogrammed on a set $\mathcal{S} \subset \mathcal{R}$ of inputs, the probability of an adversary $\mathcal{A}$ behaving differently can be related to the success probability of an extractor algorithm $\mathcal{B}$ which extracts at least one element of $\mathcal{S}$ by measuring the query register of one of $\mathcal{A}$'s randomly chosen oracle queries.

Specifically, we use the semi-classical variant of the oneway-to-hiding lemma from [5]. We recall the definition of semi-classical oracles in Definition 1 and the lemma itself in Theorem 1.

Definition 1 (Semi-Classical and Punctured Oracles). *Let $f : \mathcal{X} \to \{0, 1\}$ be a function and $\mathcal{S} \subset \mathcal{X}$ a subset s.t. $f(x) = 1$ if and only if $x \in S$. The semi-classical oracle O_f^{SC} (or equivalently $\mathrm{O}_{\mathcal{S}}^{SC}$)*

is defined as the composition of the unitary U_f and a measurement of the output register in the standard basis.

Let $\mathcal{A}^{\mathsf{O}_f^{SC}}(z)$ be a quantum algorithm which gets arbitrary input z and access to O_f^{SC}. We call the event that $\mathcal{A}$ queries O_f^{SC} on an input that yields 1 FIND.

Let H be another quantum oracle with domain $\mathcal{X}$ and some codomain Y. We define the punctured *oracle $H \setminus \mathcal{S}$ as a quantum oracle that first runs $\mathsf{O}_{\mathcal{S}}^{SC}$ and then H on the result.*

Note that as long as FIND does not occur, H and $H \setminus \mathcal{S}$ behave identically.

QUERY DEPTH AND QUERY PARALLELISM. Following [5] we are going to consider the query depth d of an adversary making in total q random oracle queries. This is important in practice since for highly-parallel adversaries we have $d \ll q$. We obtain the bounds for sequential adversaries by setting $d := q$.

Theorem 1 (Semi-Classical O2H ([5], Theorem 1)). *Let $\mathcal{S} \subset \mathcal{R}$ be random. Let G, H be random functions satisfying $\forall r \notin \mathcal{S} : \mathsf{G}(r) = \mathsf{H}(r)$. Let z be a random classical value. ($\mathcal{S}, \mathsf{G}, \mathsf{H}, z$ may have arbitrary joint distribution.) Let $\mathcal{A}$ be a quantum oracle algorithm with query depth d, expecting input z and*

$$P_{left} := \Pr\left[b = 1 \mid b \leftarrow \mathcal{A}^{\mathsf{G}}(z)\right]$$
$$P_{right} := \Pr\left[b = 1 \mid b \leftarrow \mathcal{A}^{\mathsf{H}}(z)\right]$$
$$P_{find} := \Pr\left[\textit{FIND} \,\middle|\, \mathcal{A}^{\mathsf{G}\setminus\mathcal{S}}\right]$$

Then

$$|P_{left} - P_{right}| \leq 2\sqrt{(d+1)P_{Find}} \tag{1}$$

and

$$|\sqrt{P_{left}} - \sqrt{P_{right}}| \leq 2\sqrt{(d+1)P_{Find}}\ . \tag{2}$$

3 Group Actions

In this section, we introduce cyclic effective group actions (with twists). We explain that this abstract framework models the group action underlying the isogeny-based protocol CSIDH [15]. Further we define the two standard group action assumptions in our setting: Group Action Discrete Logarithm Problem and the Group Action Computational Diffie-Hellman Problem.

3.1 Definitions

We first recall the definition of (effective) group actions from [3] and then introduce cyclic effective group actions (with twists).

Definition 2 (Group Action). *Let $(\mathcal{G}, \circ)$ be a group with identity element $e \in \mathcal{G}$, and $\mathcal{X}$ a set. A map*

$$\star : \mathcal{G} \times \mathcal{X} \rightarrow \mathcal{X}$$

is a group action if it satisfies the following properties:
1. *Identity: $e \star x = x$ for all $x \in \mathcal{X}$.*
2. *Compatibility: $(g \circ h) \star x = g \star (h \star x)$ for all $g, h \in \mathcal{G}$ and $x \in \mathcal{X}$.*

We use the shorthand notation $(\mathcal{G}, \mathcal{X}, \star)$ to denote the group action. A group action is called regular*, if for all $x, y \in \mathcal{X}$, there exists precisely one $g \in \mathcal{G}$ s.t. $y = g \star x$.*

Example 1. Let $(\mathbb{G}, \circ)$ be a group of prime order p. Then the group $(\mathbb{Z}_p^*, \cdot)$ acts on $\mathbb{G}$ in a natural way:

$$\begin{aligned} \star : \mathbb{Z}_p^* \times \mathbb{G} &\rightarrow \mathbb{G}, \\ (a, g) &\mapsto g^a = \underbrace{(g \circ \cdots \circ g)}_{a \text{ times}}. \end{aligned}$$

The identity property is trivially satisfied: $1 \star g = g^1 = g$. Compatibility is verified as follows:

$$(a \cdot b) \star g = g^{a \cdot b} = (g^a)^b = a \star (b \star g).$$

Note that the group action is not regular since $g^a \neq e \in \mathbb{G}$ for all $a \in \mathbb{Z}_p^*$. However it can be easily made regular by restricting the action to the set $\mathcal{X} = \mathbb{G} \setminus \{e\}$.

We would like to stress that exponentiation does *not* define a group action of $(\mathbb{Z}_p, +)$ on $\mathbb{G}$, since consecutive exponentiation behaves multiplicatively.

Definition 3 (Effective Group Action). *Let $(\mathcal{G}, \mathcal{X}, \star)$ be a group action satisfying the following properties:*

1. *$\mathcal{G}$ is finite and there exist efficient algorithms for membership testing, equality testing, (random) sampling, group operation and inversion.*
2. *The set $\mathcal{X}$ is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element $\tilde{x} \in \mathcal{X}$ with known representation.*
4. *There exists an efficient algorithm to evaluate the group action, i.e. to compute $g \star x$ given g and x.*

Then we call $\tilde{x} \in \mathcal{X}$ the origin and $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ an effective group action (EGA).

For an EGA it is a priori not assumed that the structure of the group $\mathcal{G}$ is known. Indeed this reflects the reality for some group actions used in cryptography. However, the security of these protocols should not be based on the difficulty of computing the group structure. Consequently, in this work we focus on known-order groups (see also [3, Definition 3.9]). Moreover, we restrict our attention to cyclic groups of known order. This is not a serious restriction. For example in isogeny-based group actions, the group $\mathcal{G}$ is guaranteed to be "almost" cyclic, hence the security is dominated by its largest cyclic component (cf. Section 3.2). For completeness, we provide an alternative set of definitions for non-cyclic abelian groups in Appendix A.

Definition 4 (Cyclic Effective Group Action). *Let $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ be an effective group action satisfying the following properties:*

1. *The group $\mathcal{G}$ is cyclic of order N for some known $N \in \mathbb{N}$.*
2. *There exists a generator $g \in \mathcal{G}$ with known representation (that is $\mathcal{G} = \langle g \rangle$).*
3. *For any element $h \in \mathcal{G}$, the element $\mathfrak{a} \in \mathbb{Z}_N$ satisfying $h = g^{\mathfrak{a}}$ is efficiently computable.*
4. *The group action is regular.*

Then we say that $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ is a cyclic (known-order) effective group action (CEGA). *In other words, a* CEGA *is an* EGA *for which there exists an isomorphism $\phi : (\mathbb{Z}_N, +) \to (\mathcal{G}, \circ)$ efficiently computable in both directions. We therefore denote any* CEGA *equivalently by $(\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ where*

$$\begin{aligned} \diamond : \mathbb{Z}_N \times \mathcal{X} &\to \mathcal{X}, \\ (\mathfrak{a}, x) &\mapsto \underbrace{(g \circ \cdots \circ g)}_{\mathfrak{a} \text{ times}} \star x\,. \end{aligned}$$

Remark 1. We stress that for a CEGA the compatibility property of a group action turns into $\mathfrak{a} \diamond (\mathfrak{b} \diamond x) = (\mathfrak{a} + \mathfrak{b}) \diamond x$.

Similar to the framework suggested in [1], we also introduce CEGA with twists which reflects a property inherent to CSIDH-based group actions.

Definition 5 (Cyclic Effective Group Action with Twists). *Let $(\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ be a* CEGA. *We call it a Cyclic Effective Group Action with Twists* (CEGAT) *if there is an efficient algorithm that, given $x = \mathfrak{a} \diamond \tilde{x}$, computes $x^t = -\mathfrak{a} \diamond \tilde{x}$.*

Remark 2. In practice, the requirements from the definition of EGA are often too strong. Therefore the weaker notion of *restricted effective group actions* (REGA) is introduced in [3]. In the design of protocols it is important to have this limitation in mind. For instance, the CSIDH protocol [15] is modeled as a REGA. For certain CSIDH parameter sets, the group structure has already been computed in [10] which converts it into a known-order REGA. Moreover the authors show that in this case it can even be modeled as a known-order EGA.[2]

In any case, in this work we are only concerned with analyzing the security of protocols. It is clear that the security should not be based on the difficulty of computing the group structure. Even more, knowing the group structure only makes a potential adversary stronger. Consequently, it makes sense to only consider *known-order effective group actions.*

3.2 The CSIDH group action

An important example of group actions used in cryptography are provided by isogeny-based group actions, in particular by CSIDH [15].

Let p be a large prime of the form $p = 4 \cdot \ell_1 \cdots \ell_n - 1$, where the ℓ_i are small distinct odd primes. Fix the elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$. This is a supersingular curve and its $\mathbb{F}_p$-rational endomorphism ring is $\mathcal{O} = \mathbb{Z}[\pi]$, where π is the Frobenius endomorphism. Let $\mathcal{E}\ell\ell_p(\mathcal{O})$ be the set of elliptic curves defined over $\mathbb{F}_p$, with endomorphism ring $\mathcal{O}$. The ideal class group $cl(\mathcal{O})$ acts on the set $\mathcal{E}\ell\ell_p(\mathcal{O})$, i.e. there is a map

$$\star : cl(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}) \to \mathcal{E}\ell\ell_p(\mathcal{O})$$

satisfying the properties from Definition 2 [15, Theorem 7].

CSIDH-512 is a CEGAT. The structure of the group $cl(\mathcal{O})$ is unknown for large parameter sets. Assuming the Generalized Riemann Hypothesis, it can be computed in classical subexponential time using the Hafner–McCurley algorithm [25]. Moreover there exist quantum algorithms by Biasse and Song [11] to compute the class group $cl(\mathcal{O})$ in polynomial time. The class group computation has been successfully performed for the parameters of CSIDH-512 [10]. In particular, the authors showed that $cl(\mathcal{O})$ is a cyclic group of order

$$\begin{aligned} N &= 3 \cdot 37 \cdot 1407181 \cdot 51593604295295867744293584889 \\ &\quad \cdot 31599414504681995853008278745587832204909 \\ &\approx 2^{257} \end{aligned} \tag{3}$$

generated by the element $g = (3, \pi - 1) \in cl(\mathcal{O})$. Moreover it is shown that $cl(\mathcal{O})$ can be identified with $\mathbb{Z}_N$ via an efficiently computable isomorphism

$$\phi : \mathbb{Z}_N \to cl(\mathcal{O}), \quad \mathfrak{a} \mapsto g^{\mathfrak{a}}.$$

Consequently, the CSIDH-512 group action can be viewed as a CEGA. Moreover as any CSIDH-based group action, it is possible to compute twists of elements efficiently, hence it may also be viewed as a CEGAT.

Remark 3. While not all class groups appearing in isogeny-based cryptography are cyclic, it makes sense to model these groups as CEGATs when analyzing their security. This is justified by the fact that heuristically the class groups are close to being cyclic, see also [15, §7.1]. More precisely, the odd part of a randomly chosen class group of an imaginary quadratic field is very likely to be cyclic. In case it is not cyclic, the group is with overwhelming probability of the form $\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$ with $N_2 \ll N_1$, hence its complexity is dominated by the cyclic component $\mathbb{Z}_{N_1}$. More details on these heuristics can be found in [17, §9.1]. Further, we note that genus theory implies that the even part of $cl(\mathcal{O})$ is trivial for all CSIDH parameters.

[2] As is pointed out in [3], knowing the group structure of a REGA does not automatically covert it to an EGA, but there are some subtleties to consider. In particular, the evaluation of the group action might require to solve a lattice problem. However, in all known instantiations this problem is easy to solve [10].

3.3 Group Action Assumptions

For cryptographic applications, we are interested in $\mathsf{CEGA(T)}$s that come equipped with the following two properties:

– Given $x \in \mathcal{X}$, it is hard to find $\mathfrak{a} \in \mathbb{Z}_N$ with $\mathfrak{a} \diamond \tilde{x} = x$.
– Given $x = \mathfrak{a} \diamond \tilde{x},\ y = \mathfrak{b} \diamond \tilde{x}$, it is hard to find $z \in \mathcal{X}$ with $z = (\mathfrak{a} + \mathfrak{b}) \diamond \tilde{x}$.

In [3] such group actions are called *cryptographic group actions*, and in [18] they are called *hard homogeneous spaces*. The two hardness assumptions are the natural generalizations of the discrete logarithm assumption and the Diffie-Hellman assumption in the traditional group based setting. In analogy to this setting, we make the following definitions.

Definition 6 (Group Action Discrete Logarithm Problem). *Let* $\mathsf{CEGA(T)} = (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ *be a cyclic effective group action (with twists). We say that an adversary* $\mathcal{A}$ *solves the group action discrete logarithm problem* ($\mathsf{GA\text{-}DLOG}$) *if* $\mathcal{A}(\tilde{x}, \mathfrak{a} \diamond \tilde{x}) = \mathfrak{a}$ *for* $\mathfrak{a} \xleftarrow{\$} \mathbb{Z}_N$.

Definition 7 (Group Action Computational Diffie-Hellman Problem). *Let* $\mathsf{CEGA(T)} = (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ *be a cyclic effective group action (with twists). We say that an adversary* $\mathcal{A}$ *solves the group action computational Diffie-Hellman problem* ($\mathsf{GA\text{-}CDH}$) *if* $\mathcal{A}(\tilde{x}, \mathfrak{a} \diamond \tilde{x}, \mathfrak{b} \diamond \tilde{x}) = (\mathfrak{a} + \mathfrak{b}) \diamond \tilde{x}$ *for* $\mathfrak{a}, \mathfrak{b} \xleftarrow{\$} \mathbb{Z}_N$.

4 Generic Group Action Model

In this section we adapt the well-known Generic Group Model (GGM) of Shoup [42] to the group action setting. In Section 4.1 we introduce the Generic Group Action Model and relate it to the GGM. We extend the definition to also include twists in Section 4.2 and quantum queries in Section 4.3. There, we also show that $\mathsf{GA\text{-}DLOG}$ does not hold when allowing quantum queries. Finally, we compare our model to the model of [35] in Section 4.4.

As explained in the preceding section, we focus on CEGAs for the underlying group action of our generic models. For completeness a natural extension to known-order effective group actions is explained in Appendix A.

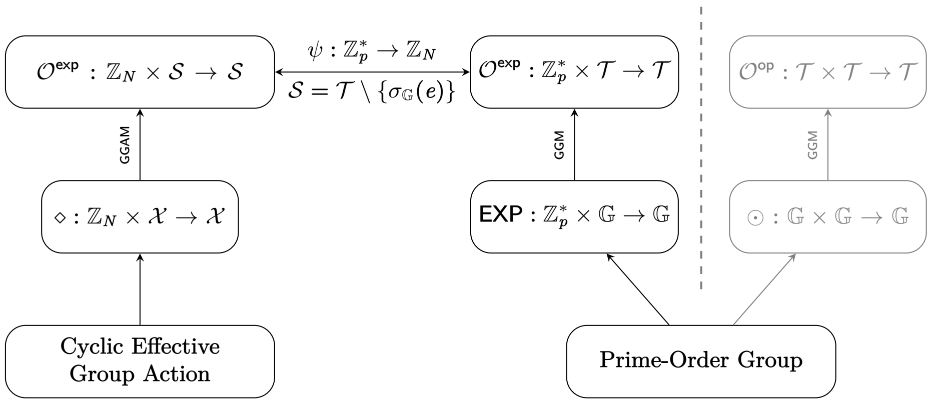
4.1 Definitions and Relations

Generic Group Model. Recall that in the GGM we associate to a group $(\mathbb{G}, \odot, e)$ of prime order p and neutral element e a set of labels $\mathcal{T} \subset \{0,1\}^*$ and an injective labeling function $\sigma_{\mathbb{G}} : \mathbb{G} \to \mathcal{T}$. We require $|\mathcal{T}| = |\mathbb{G}|$, but we assume that labels are sufficiently long to ensure that finding a label that has a corresponding preimage in $\mathbb{G}$ is hard. A generic algorithm is subsequently given abstract access to $\mathbb{G}$ via two oracles $\mathcal{O}^{\mathsf{op}} : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$ and $\mathcal{O}^{\mathsf{exp}} : \mathbb{Z}_p^* \times \mathcal{T} \to \mathcal{T}$.[3] The first oracle takes as input two labels $\sigma_{\mathbb{G}}(g)$ and $\sigma_{\mathbb{G}}(h)$ and returns $\sigma_{\mathbb{G}}(g \odot h)$. Similarly, the second oracle takes as input a label $\sigma_{\mathbb{G}}(g)$ and an exponent $k \in \mathbb{Z}_p^*$ and returns $\sigma_{\mathbb{G}}(g^k)$. Here we restrict the exponent k to be in $\mathbb{Z}_p^*$ instead of $\mathbb{Z}_p$. This restriction is without loss of generality, as $\mathcal{O}^{\mathsf{exp}}(0, \sigma_{\mathbb{G}}(g)) = \sigma_{\mathbb{G}}(e)$ for all $g \in \mathbb{G}$, which is trivial to compute even without the oracle. However, it allows us to view $\mathcal{O}^{\mathsf{exp}}$ as the *regular* group action shown in Example 1. Lastly, we call p the *order* of the GGM.

Let $\mathcal{A}$ be a generic algorithm playing a game G. We say $\mathcal{A}$ is a (ϵ, t, q)-algorithm if its has success probability $\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] = \epsilon$, time complexity t and query complexity q. Here, the query complexity refers to the amount of queries to $\mathcal{O}^{\mathsf{exp}}$ and $\mathcal{O}^{\mathsf{op}}$. We call a generic (ϵ, t, q)-algorithm *pseudoefficient* if $q \in \mathcal{O}(\mathit{poly}(\log p))$ and simply *efficient* if both $t, q \in \mathcal{O}(\mathit{poly}(\log p))$.

Generic Group Action Model (GGAM). The Generic Group Action Model is now defined similarly to the GGM. We associate to a cyclic effective group action $\mathsf{CEGA} = (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ a set of labels $\mathcal{S} \subset \{0,1\}^*$ and an injective labeling function $\sigma_{\mathcal{X}} : \mathcal{X} \to \mathcal{S}$. We again assume that $\mathcal{X}$ and $\mathcal{S}$ have the same cardinality. Subsequently, a generic group action algorithm has access to an oracle $\mathcal{O}^{\mathsf{exp}} : \mathbb{Z}_N \times \mathcal{S} \to \mathcal{S}$ which abstractly computes the group action. In particular, on input $\mathfrak{a} \in \mathbb{Z}_N$ and a label $\sigma_{\mathcal{X}}(x)$ for $x \in \mathcal{X}$, the oracle returns $\sigma_{\mathcal{X}}(\mathfrak{a} \diamond x)$. While this is slightly ambiguous with the $\mathcal{O}^{\mathsf{exp}}$ in the GGM, it is easy

[3] We include $\mathcal{O}^{\mathsf{exp}}$ as a distinct oracle for convenience even though it can be simulated efficiently via $\mathcal{O}^{\mathsf{op}}$ and a square-and-multiply approach.



$\mathcal{O}^{\mathsf{exp}} : \mathbb{Z}_N \times \mathcal{S} \to \mathcal{S}$
$\psi : \mathbb{Z}_p^* \to \mathbb{Z}_N$
$\mathcal{S} = \mathcal{T} \setminus \{\sigma_{\mathbb{G}}(e)\}$
$\mathcal{O}^{\mathsf{exp}} : \mathbb{Z}_p^* \times \mathcal{T} \to \mathcal{T}$
$\mathcal{O}^{\mathsf{op}} : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$
GGAM
GGM
GGM
$\diamond : \mathbb{Z}_N \times \mathcal{X} \to \mathcal{X}$
$\mathsf{EXP} : \mathbb{Z}_p^* \times \mathbb{G} \to \mathbb{G}$
$\odot : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$
Cyclic Effective Group Action
Prime-Order Group

Fig. 1. Schematic overview of the relation between GGAM and GGM where $N = p - 1$.

to see that the $\mathcal{O}^{\mathsf{exp}}$ in the GGM and the $\mathcal{O}^{\mathsf{exp}}$ in the GGAM are virtually identical and, as we will argue next, can be translated into one another. We call N the *order* of the GGAM. Lastly, a generic (ϵ, t, q)-algorithm playing a game G is called pseudoefficient if $q \in \mathcal{O}(poly(\log N))$ and overall efficient if $t, q \in \mathcal{O}(poly(\log N))$.

One can now observe that the definition of the GGAM almost matches the definition of the GGM except for the $\mathcal{O}^{\mathsf{op}}$ oracle. In fact, the only other difference between both models is the fact that in the GGM, $\mathcal{O}^{\mathsf{exp}}$ takes as input an exponent in $\mathbb{Z}_p^*$ whereas in the GGAM, the exponents are in $\mathbb{Z}_N$. Assuming $N = p - 1$, this change is w.l.o.g. as there exists an isomorphism $\psi : \mathbb{Z}_p^* \to \mathbb{Z}_N$ which a (pseudoefficient) generic algorithm can easily compute. Also note that $\mathcal{S}$ and $\mathcal{T}$ can be identified except for the label $\sigma_{\mathbb{G}}(e) = \sigma_{\mathbb{G}}(g^0)$ since $0 \notin \mathbb{Z}_p^*$, but this label can simply be ignored. The relation between the GGM and the GGAM is visualized in Figure 1.

From the construction of the GGAM we easily see that it is a stronger (i.e. more restricted) version of the GGM if we assume $N + 1$ being prime. We therefore get the following observation.

Lemma 1 (Lifting Lemma). *Let p be a prime and define $N = p - 1$. Let* G *be a game in the GGAM of order N and let $\mathcal{A}$ be a generic (ϵ, t, q)-algorithm winning* G *in the GGAM. Then there exists a generic (ϵ, t', q)-algorithm $\mathcal{B}$ that wins* G *in the GGM of order p with runtime $t' \leq q + t$.*

Proof. From the observations above, it is straight forward to translate the algorithm $\mathcal{A}$ in the GGAM to an algorithm $\mathcal{B}$ in the GGM: Because the label sets $\mathcal{S}$ and $\mathcal{T}$ can be identified except for $\sigma_{\mathbb{G}}(e)$, we just have to worry about translating oracle queries between the GGAM and GGM. As Figure 1 shows, this boils down to computing the isomorphism ψ in the backwards direction, i.e. computing $\psi^{-1} : \mathbb{Z}_N \to \mathbb{Z}_p^*$. This can be done efficiently via $\mathfrak{a} \mapsto g^{\mathfrak{a}}$ for some generator $g \in \mathbb{Z}_p^*$. Therefore the runtime of $\mathcal{B}$ increases by the number of evaluations of ψ^{-1}, which itself is exactly the amount q of oracle queries that $\mathcal{A}$ issues. □

Remark 4. Note that when viewing G in the GGM we might lose some (trivial) instances. More specifically, the game G in Lemma 1 is defined in the GGAM first, which means that the label $\sigma_{\mathbb{G}}(e)$ cannot occur in an instance of G as it is not defined in the GGAM. For example, in the case of DLOG this means that the instance $(g, g^0) = (g, e)$ cannot occur in the GGM as $\psi(0)$ is undefined and therefore this does not have a corresponding instance in GA-DLOG. Yet this is only a formal restriction that does not affect the hardness of G in the GGM.

Intuitively, Lemma 1 states that hardness result in the GGM can be lifted to the GGAM as long as $N = p - 1$. Most importantly the lemma also applies to the hardness of solving GA-DLOG and GA-CDH [42].

Corollary 1. *Assume $N + 1$ being prime. For a generic (ϵ, t, q)-algorithm solving either* GA-DLOG *or* GA-CDH *in the GGAM of order N we have $\epsilon \leq q^2/N$.*

Note that the *Baby-Step Giant-Step Algorithm* [40] can be easily adapted to the group action setting. This adaption yields a generic algorithm for solving GA-DLOG which matches the bound from Corollary 1.

We remark again that Lemma 1 and Corollary 1 only apply if $N+1$ is a prime number. Therefore hardness results in the GGM only carry over to the GGAM under this restriction. For specific assumptions we can prove their hardness irrespective of $N+1$ being prime as we show in the next section, however a universal lifting theorem might not be achievable. We leave this as an interesting topic for future work.

4.2 Generic Group Action Model with Twists

We introduce the Generic Group Action Model with Twists (GGAM$^\top$) which extends the above GGAM for CEGATs. Recall that current instantiations of group actions like CSIDH are indeed modeled more accurately by a CEGAT as they come with an additional twisting functionality which allows to efficiently compute $-\mathfrak{a} \diamond \tilde{x}$ from $\mathfrak{a} \diamond \tilde{x}$. Since this functionality is not present in the current definition of the GGAM, some attacks may not be captured for a large class of widely used group actions. This makes an important difference when analyzing the generic hardness of non-standard assumptions. It is therefore desirable to adapt the GGAM to this setting. In particular, we extend the definition of the GGAM by an additional oracle $\mathcal{O}^{\mathsf{tw}} : \mathcal{S} \to \mathcal{S}$ that computes $\sigma_{\mathcal{X}}(-\mathfrak{a} \diamond \tilde{x})$ for a given input $\sigma_{\mathcal{X}}(\mathfrak{a} \diamond \tilde{x})$.

Note that twists do not have an analogue in the standard prime order group setting. This means that Lemma 1 does not apply to the GGAM$^\top$ even if $N+1$ is prime. Below we provide two separating examples.

Example 2. The security of the password-autheticated key exchange scheme TBPEKE [37] relies on the so-called *Simultaneous Diffie-Hellman assumption.* Translated to the group action notation, an adversary is given three elements $x = \mathfrak{g} \diamond \tilde{x},\ y_1 = \mathfrak{a}_1 \diamond x,\ y_2 = \mathfrak{a}_2 \diamond x \in \mathcal{X}$ and is supposed to output three elements $z,\ r_1,\ r_2 \in \mathcal{X}$ satisfying $r_1 = -\mathfrak{a}_1 \diamond z$ and $r_2 = -\mathfrak{a}_2 \diamond z$. The authors show that a generic (ϵ, t, q)-algorithm solving this problem in the GGM of order p, has success probability $\epsilon \leq q^2 + 20/2p$. In contrast to that, an adversary in the GGAM$^\top$ only needs three calls to the $\mathcal{O}^{\mathsf{tw}}$ oracle in order to find a valid solution: $z = \mathcal{O}^{\mathsf{tw}}(x),\ r_1 = \mathcal{O}^{\mathsf{tw}}(y_1),\ r_2 = \mathcal{O}^{\mathsf{tw}}(y_2)$.

Example 3. Another simple separating example is the *Inverse Diffie-Hellman Problem (IDHP).* Adapted to our group action notation, an adversary is given $\mathfrak{a} \diamond \tilde{x}$ for $\mathfrak{a} \xleftarrow{\$} \mathbb{Z}_N$ and is asked to compute $-\mathfrak{a} \diamond \tilde{x}$. From the definition of $\mathcal{O}^{\mathsf{tw}}$ it is obvious that IDHP is easy in GGAM$^\top$. In the GGAM, however, IDHP must be as hard as CDH (assuming $N+1$ being prime). The reason is that IDHP and CDH are equivalent in the GGM [7], implying that both have the same lower bound in the GGM. Due to Lemma 1, the lower bound for IDHP must therefore hold in the GGAM. We thus have that IDHP is easy in the GGAM$^\top$ but provably hard in the GGAM if $N+1$ is prime.

The two examples show that results in the GGAM$^\top$ and the GGAM are incomparable. However, we can still analyze specific problems directly in the GGAM$^\top$ by adapting the information theoretic arguments used in the GGM. With this we get a similar bound for the discrete logarithm assumption, as well as bounds for the non-standard Group Action k-power Decisional Diffie-Hellman Problem (Definition 8) and the Group Action Discrete Logarithm Problem with Auxilary Input (Definition 9). Our analysis further shows that some assumptions possess instances which are potentially easier to solve. We are further able to argue that information-theoretically there are only a hand full of these cases which are all linked to the composite nature of N. Before we proceed to study these problems, we recall a useful lemma needed for our analysis.

Lemma 2. *Let $a, b \in \mathbb{Z}$, $N \in \mathbb{N}$ and denote $d = \gcd(a, N)$. Then the equation $a \cdot x \equiv b \pmod N$ has precisely d solutions if d divides b and no solutions otherwise.*

Theorem 2. *For every generic (ϵ, t, q)-algorithm $\mathcal{A}$ winning the* GA-DLOG *game in the GGAM$^\top$ of order N, we have $\epsilon \leq 2q^2/N$.*

Note that the bound differs from the GGM bound by a factor of 2. This is due to the fact that N is potentially composite, leading to polynomials of degree 1 with two roots in the reduction. This factor can be removed by requiring that $\gcd(2, N) = 1$. The latter is always true when instantiating the group action with CSIDH.

Proof. The proof idea is very similar to classical proofs of DLOG in the GGM. The discrete logarithm challenge is replaced by an indeterminate $\mathbf{X}$ and the labeling function is extended to polynomials $f_i(\mathbf{X})$. If the value of $\mathbf{X}$ (and therefore the DLOG solution) is only chosen after the adversary finished, the probability of success is exactly $1/N$. However, we have to ensure that this change is undetected. The only problem that can occur is that the adversary makes two queries (which are now on polynomials $f_i(\mathbf{X}), f_j(\mathbf{X})$) that result in different labels since $f_i(\mathbf{X}) \neq f_j(\mathbf{X})$, but both polynomials evaluate to the same value on the chosen challenge. However, we can bound the probability of this event by predicting the number of roots of $f_i(\mathbf{X}) - f_j(\mathbf{X})$, yielding the well-known bounds.

While our proof follows the same idea, there are some intricacies we have to consider. First, the order N of the group action is potentially composite, so by Lemma 2, for a polynomial of the form $\mathfrak{a}\mathbf{X} + \mathfrak{b} \equiv 0 \pmod N$, there are $d = \gcd(\mathfrak{a}, N)$ many roots if d divides $\mathfrak{b}$ and none otherwise. On the other hand, an adversary is limited to the $\mathcal{O}^{\mathsf{exp}}$ and the $\mathcal{O}^{\mathsf{tw}}$, so all equations that it can compute are of the form $\mathfrak{g} \pm \mathbf{X}$ or $\mathfrak{g} \pm 2\mathbf{X}$, thus every equation has at most two roots.

Overall with exactly the same argument as in the GGM the theorem follows with the additional factor 2. ◻

In [19] the authors define the so called *k-power Decisional Diffie-Hellman Group Action Problem* in order to build threshold schemes. We give an equivalent definition that is written in additive notation and analyze its hardness in the GGAM$^{\top}$.

Definition 8 (Group Action k-power Decisional Diffie-Hellman Problem). *Let* $\mathsf{CEGAT} = (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ *and* $1 < k < N-1$. *For* $\mathfrak{a}, \mathfrak{c} \xleftarrow{\$} \mathbb{Z}_N$ *define* $w_0 = k \cdot \mathfrak{a} \diamond \tilde{x}$ *and* $w_1 = \mathfrak{c} \diamond \tilde{x}$. *We say that an adversary* $\mathcal{A}$ *solves the Group Action k-power Decisional Diffie-Hellman Problem* (GA-k-PDDH) *if*

$$\mathcal{A}(k, \mathfrak{a} \diamond \tilde{x}, w_b, w_{1-b}) = b$$

for $b \xleftarrow{\$} \{0,1\}$.

Theorem 3. *For every generic* (ϵ, t, q)*-algorithm* $\mathcal{A}$ *winning* GA-k-PDDH *in the GGAM*$^{\top}$ *of order* N*, we have*

$$\epsilon \leq \frac{1}{2} + \frac{2 \cdot d_{max} \cdot q^2}{N} ,$$

where $d_{max} = \max\{\gcd(2k, N), \gcd(k \pm 1, N)\}$.

Proof. The proof uses an argument similar to the original proof of DDH in the GGM by Shoup [42], however there are again some subtleties. Fix some $1 < k < N-1$. We define indeterminates $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ that represent the values $\mathfrak{a} \diamond \tilde{x}$, w_b and w_{1-b}, respectively. An algorithm can implicitly build polynomials $f_i(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ of the following forms:

$$\mathfrak{g}, \qquad \mathfrak{g} \pm \mathbf{X}, \qquad \mathfrak{g} \pm \mathbf{Y}, \qquad \text{and} \qquad \mathfrak{g} \pm \mathbf{Z} \qquad \pmod N .$$

As in the proof of Theorem 2 we are now interested in the number of roots of $f_i(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) - f_j(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$. More specifically, we have to keep in mind that either $\mathbf{Y}$ or $\mathbf{Z}$ is set to $k \cdot \mathbf{X}$ at the end of the game, meaning that we have to bound the probability that either

$$f_i(\mathbf{X}, \mathbf{Y}, k \cdot \mathbf{X}) - f_j(\mathbf{X}, \mathbf{Y}, k \cdot \mathbf{X}) \equiv 0 \mod N \tag{4}$$

or

$$f_i(\mathbf{X}, k \cdot \mathbf{X}, \mathbf{Z}) - f_j(\mathbf{X}, k \cdot \mathbf{X}, \mathbf{Z}) \equiv 0 \mod N \tag{5}$$

for a random assignment of $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. Listing all possibilities for these differences is tedious, therefore we restrict our attention to the polynomial in $\mathbf{X}$ and leave the remaining polynomials (in either $\mathbf{Y}$ or $\mathbf{Z}$) unspecified. In the case of eq. (4) we get

$$\pm k \cdot \mathbf{X} + f(\mathbf{Y}), \quad \pm(k+1)\mathbf{X} + f(\mathbf{Y}), \quad \pm(k-1)\mathbf{X} + f(\mathbf{Y}),$$

$$\pm \mathbf{X} + f(\mathbf{Y}), \quad \pm 2\mathbf{X} + f(\mathbf{Y}), \quad \pm 2k \cdot \mathbf{X} + f(\mathbf{Y}) .$$

and analogously for eq. (5). The likelihood of an assignment being a root now mainly depends on the choice of $\mathbf{X}$. Recall from Lemma 2 that an equation of the form $\mathfrak{a}\mathbf{X} + \mathfrak{b} \equiv 0 \pmod{N}$ has at most $\gcd(\mathfrak{a}, N)$ solutions. In our setting, $\mathfrak{a} \in \{\pm 1, \pm 2, \pm k, \pm(k-1), \pm(k+1), \pm 2k\}$, hence

$$\gcd(\mathfrak{a}, N) \leq d_{\max} = \max\{\gcd(2k, N), \gcd(k \pm 1, N)\} \ .$$

In conclusion, the probability that an assignment for $\mathbf{X}$ is a root of one polynomial in eq. (4) is bounded by $d_{\max}/N$ and the same holds for polynomials in eq. (5). Therefore, choosing the bit b after the adversary has finished fails with probability $2 \cdot d_{\max}/N$. Of course, an adversary can still guess the (now information theoretically hidden) bit b with constant probability $1/2$, which immediately yields the overall bound. □

Remark 5. The authors in [19] already noted that in the presence of twists GA-k-PDDH is easy for $k = N - 1$. Generically, the problem can be solved by sending one query to the twist oracle. Therefore we excluded this case in our definition of the problem. Moreover our analysis in the GGAM$^{\top}$ indicates that there are more choices for $1 < k < N-1$ for which GA-k-PDDH becomes potentially easier. In light of Theorem 3, we advise to only use a stronger version of GA-k-PDDH, where it is additionally assumed that $d_{\max} = \max\{\gcd(2k, N), \gcd(k \pm 1, N)\} = 1$.

Remark 6. It is also possible to consider a version of GA-k-PDDH where all elements are given relative to some $x = \mathfrak{h} \diamond \tilde{x}$. A correct GA-$k$-PDDH tuple would thus contain the element $z = (k \cdot \mathfrak{a} + \mathfrak{h}) \diamond \tilde{x}$. This makes the analysis more intricate but does not change the overall bound.

Baghery, Cozzo and Pederson [6] introduce the vectorization problem with auxiliary inputs to construct a new identification scheme and thus a signature scheme based on CSIDH. The problem has already been analyzed by [28] in the standard model. It is similar to q-DLOG [12] or DLOG with auxiliary inputs [16] in the prime-order group setting. In the following we first recall the problem and then analyze its hardness in the GGAM$^{\top}$.

Definition 9 (Group Action Discrete Logarithm Problem with Auxilary Input [6]). *Let* CEGAT $= (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$. *We say that an adversary $\mathcal{A}$ solves the Group Action Discrete Logarithm Problem with Auxilary Input* (GA-DLAI) *if*

$$\mathcal{A}(k_0, \ldots, k_{m-1}, x_0, \ldots, x_{m-1}) = \mathfrak{a}$$

where the k_i are given by

$$k_0 = 0, k_1 = 1 \textit{ and } 1 < k_i < N \textit{ for } i > 1$$

under the restriction

$$\forall i \neq j : \gcd(k_i - k_j, N) = 1 \qquad \textit{and} \qquad \forall i, j : \gcd(k_i + k_j, N) = 1$$

and the x_i are given by $x_i = k_i \cdot \mathfrak{a} \diamond \tilde{x}$ for $\mathfrak{a} \xleftarrow{\$} \mathbb{Z}_N$.

Theorem 4. *For any generic (ϵ, q, t)-adversary $\mathcal{A}$ winning* GA-DLAI *in the GGAM$^{\top}$ of order N, we have $\epsilon \leq q^2/N$.*

Proof. The proof is almost identical to the proofs of Theorem 2 and Theorem 3. Here, the cases where an equation has more than one root are excluded by the requirement that $\gcd(k_i - k_j, N) = 1$ for $i \neq j$ and $\gcd(k_i + k_j, N) = 1$ for all i, j. □

4.3 Quantum Generic Group Action Model

Isogeny-based group actions are a strong candidate for post-quantum secure cryptography. Therefore, group action based assumptions should also be analyzed in the presence of quantum adversaries. One might hope that the GGAM can be adapted to the quantum setting where a quantum algorithm can query $\mathcal{O}^{\mathsf{exp}}$ in superposition. Formally this change is simple, that is, we define the quantum oracle

$\mathcal{O}^{\mathsf{exp}}(\sum_{x,y,z} \alpha_{x,y,z} |x, y, z\rangle) = \sum_{x,y,z} \alpha_{x,y,z} |x, y, z \oplus \mathcal{O}^{\mathsf{exp}}(x, y)\rangle$[4]. In the following, we will denote this model by QGGAM and naturally extend the notion of a (ϵ, t, q)-algorithm to the quantum setting. In practice, however, making the oracle quantum accessible creates a lot of problems, especially regarding one of the main purposes of the GGAM: information-theoretic lower bounds.

The most glaring issue is an algorithm by Ettinger and Høyer [22] that can be used to solve GA-DLOG pseudoefficiently in the QGGAM.

Theorem 5. *There exists a generic quantum (ϵ, t, q)-algorithm that solves* GA-DLOG *in the QGGAM of order N with $\epsilon = 1 - \frac{1}{2N}$, $t \in \mathcal{O}(\sqrt{N})$ and $q = \lceil 64 \ln N \rceil$.*

In its purest form the Ettinger-Høyer algorithm solves the well-known *Dihedral Hidden Subgroup Problem (DHSP)* with polynomially many quantum oracle queries (see [29] for a survey on the DHSP). Because GA-DLOG and many other group action based assumptions reduce to some form of the DHSP, its hardness plays a central role in group action based cryptography [36]. Although there have been improvements towards solving the DHSP more efficiently, the currently fastest (generic) algorithm still has an overall subexponential complexity.

Theorem 6 (Kuperberg's algorithm, [31]). *There exists a generic quantum (ϵ, t, q)-algorithm that solves* GA-DLOG *in the QGGAM of order N with $\epsilon \approx 1$ and $t, q \in \mathcal{O}(2^{\sqrt{\log N}})$.*

In the standard model, Kuberberg's algorithm has the more favorable trade off between the time complexity t and query complexity q, therefore making it the best currently known algorithm against DHSP. Because of its overall subexponential complexity, the DHSP (and by extension GA-DLOG) remain hard in the standard model, which starkly contrasts the situation in the QGGAM.

We conclude that in the QGGAM we cannot give lower bounds on computational problems because even GA-DLOG can be solved with polynomial oracle complexity. Since the exponential time of the Ettinger-Høyer algorithm stems from solving a combinatorial problem, one can assume that the hardness of GA-DLOG is thus not an algebraic one but a combinatorial one. We discuss this more in Appendix B.

4.4 Comparison with the Generic Model by Montgomery and Zhandry

In [35] the authors define a different variant of the GGAM, which we call the MZ-GGAM, where the group $\mathcal{G}$ of a group action $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ is also modeled as a generic group. Consequently, the authors define *two* labeling functions, one for $\mathcal{G}$ and one for $\mathcal{X}$, as well as corresponding oracles for the operation in $\mathcal{G}$ and for the group action. In particular, DLOG is assumed to be hard in the group $\mathcal{G}$. In the following, we want to argue that this model and especially its quantum counterpart does not capture the widely used class of *(cyclic) effective* group actions very well. To simplify the exposition, we assume that in line with our setup the group $\mathcal{G}$ is cyclic. Clearly, the discussion translates to the setting of a general abelian group as well. For more details on the functionality of our GGAM in this setting, the reader is referred to Appendix A.

For the discussion we introduce the (somewhat informal) notation $[\mathcal{X}]$ resp. $[\mathcal{G}]$ to denote that a set (resp. group) is idealized in the sense of the generic group model. That is, its elements are represented by labels and algorithms are provided with oracles to compute the group action (resp. group operation). With this notation our GGAM would be written as $(\mathbb{Z}_N, [\mathcal{X}], \diamond, \tilde{x})$, while the MZ-GGAM of [35] can be seen as $([\mathcal{G}], [\mathcal{X}], \star, \tilde{x})$.

Our discussion mainly focuses on the generic group $[\mathcal{G}]$, which is the main difference between the two models. We first cover the classical setting where we observe that the MZ-GGAM can be used to simulate a classical GGM. The reason for this is straightforward: Per definition, the MZ-GGAM provides an algorithm with a generic group $[\mathcal{G}]$. In fact, any game or algorithm in the GGM can be compiled in the MZ-GGAM by just ignoring $[\mathcal{X}]$ and solely using $[\mathcal{G}]$ for computations. We thus have that, intuitively, the GGM is fully contained in the MZ-GGAM. A more formal statement would be that the GGM is *indifferentiable* from the MZ-GGAM (see [33] for a definition of indifferentiability). We leave out this rigorous discussion in the interest of space. This connection is not desirable as the computational hardness of group actions should not stem from the hardness of the group itself but instead should stem from the

[4] This was already observed in [35]. A more thorough comparison can be found in Section 4.4

hardness of inverting the group action. Phrased differently: even if DLOG were easy in $\mathcal{G}$, we still require that GA-DLOG is hard in $(\mathcal{G}, [\mathcal{X}], \star, \tilde{x})$.

The last fact is emphasized when looking at the *quantum* setting. Here, we can easily compute an isomorphism $\psi : [\mathcal{G}] \to \mathbb{Z}_N$ via Shor's algorithm [41] for a suitable $N \in \mathbb{N}$, making the generic group $[\mathcal{G}]$ obsolete for algebraic purposes. Additionally, even constructions like the random oracle from a generic group in [44] are likely to be adaptable to the QGGAM as they only require access to a random labeling function.

Of course, the issues discussed in Section 4.3 regarding the quantum hardness of GA-DLOG also apply to the MZ-QGGAM. Hence there is no reason to insist on a generic group $[\mathcal{G}]$ as any hardness assumption imposed on $\mathcal{G}$ can be circumvented by quantum preprocessing. We therefore believe that even in the classical setting, our weaker definition models the computational complexity of (cyclic) effective group actions in a more realistic way.

5 Algebraic Group Action Model

In this section we introduce the Algebraic Group Action Model and use it to prove several results. In Section 5.1 we formally define the Algebraic Group Action Model in several variants (classic/quantum and with/without twists). In Section 5.2 we use the quantum version to reduce different non-standard assumptions to GA-DLOG.

5.1 Definition and Relations

We define the Algebraic Group Action Model (AGAM) similar to the AGM of [23] with the difference that the underlying algebraic structure is now an effective group action instead of a prime-order group. Like in the GGAM, we assume that DLOG is easy in $\mathcal{G}$ and therefore make exclusive use of cyclic effective group actions.[5]

Definition 10 ((Quantum) Algebraic Group Action Algorithms). *Let* CEGA $= (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ *be a fixed cyclic effective group action. An algorithm $\mathcal{A}$ is called* algebraic *if for each output element $y \in \mathcal{X}$ it additionally provides a representation relative to a previously received set element. Concretely, if $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$ is the list of received set elements so far, $\mathcal{A}$ additionally provides a group element $\mathfrak{a} \in \mathbb{Z}_N$ and an index $i \in \{1, \ldots, \ell\}$ such that $y = \mathfrak{a} \diamond x_i$. We denote*

$$y_{(i,\mathfrak{a})} = \mathfrak{a} \diamond x_i.$$

If an oracle is queried on some set elements, then $\mathcal{A}$ also has to provide a representation for each set element contained in that query.

Analogously, a quantum *adversary is algebraic if for every output state $|y\rangle$ it additionally outputs a* quantum *representation, i.e. a quantum state $|i, \mathfrak{a}\rangle$ s.t. $|y\rangle = |\mathfrak{a} \diamond x_i\rangle$. Similarly to the classical case, we write $|y_{(i,\mathfrak{a})}\rangle$. Note that the representation is entangled with the group element.*

Additionally, if the group action supports twists (i.e. it is a CEGAT*), we extend the representation by a bit b, indicating whether the base element was twisted before applying the group action. Formally, we then have*

$$y_{(i,\mathfrak{a},b)} = \begin{cases} \mathfrak{a} \diamond x_i & \text{if } b = 0, \\ \mathfrak{a} \diamond {x_i}^t & \text{if } b = 1. \end{cases}$$

As in the AGM we require that all auxiliary input provided to the adversary which is not in $\mathcal{X}$ does not depend on elements from $\mathcal{X}$.

Remark 7. As noted in [45,27], it is somewhat imprecise to require auxiliary inputs to be independent of set element inputs, as there might be arbitrary, information theoretically hidden set elements encoded in them. However, generally it is clear in practice whether such dependencies exist, so we do not specify this further. Additionally, all adversaries that we consider only receive set elements as inputs, so it is clear either way that there are no "hidden" set elements.

[5] To capture general abelian groups, a similar approach as described for the GGAM in Appendix A applies.

Remark 8. We assume that all representations provided by an adversary in the QAGAM are correct. While it is not as straight forward as in the classical setting to actually check this correctness, this is without loss of generality. Instead of providing the representation *and* the group element itself, we could instead require the adversary to only provide the representation and have the reduction/oracle compute the group element itself. This makes everything consistent and correct. Alternatively, an approach similar to the semi-classical O2H is possible, where the reduction measures whether the representation is correct, however this introduces a small error probability.

Let $\mathrm{G}_1, \mathrm{G}_2$ be two security games in the (Q)AGAM and assume that there exists an algebraic (ϵ, t)-algorithm $\mathcal{A}$ winning game G_2. We say that G_1 *reduces to* G_2 if there exists an *efficient* (quantum) algorithm $\mathcal{R}$ (called the *reduction*) such that $\mathcal{R}^{\mathcal{A}}$ is an (ϵ', t')-algorithm that wins G_1 with time complexity $t' \in \mathcal{O}(poly(t))$ and success probability $\epsilon' \in \Omega(poly(\epsilon))$. If $\mathcal{R}$ is algebraic, then we have that $\mathcal{R}^{\mathcal{A}}$ is an algebraic algorithm as well.

While the GGAM can not be lifted (in a useful way) to the quantum setting (see Section 4.3), the QAGAM is indeed useful for studying the relation between assumptions in the presence of a quantum adversary. Firstly, this is due to the fact that the AGAM is not used to prove lower bounds but instead upper bounds via reductions as described above. This means that all reductions from the AGAM can be immediately lifted to the QAGAM, as a quantum algorithm can perform all classical operations. Secondly, some assumptions are inherently quantum. They can for example include oracles which can be queried on quantum superpositions. For such queries, a classical representation is implausible, as it potentially has exponential size and would require an adversary to always know all amplitudes of the states it queries to oracles, which is unreasonable even for algebraic adversaries. Therefore, the QAGAM is necessary when considering such assumptions. We analyze two inherently quantum assumptions in Section 5.2.

Relation between AGAM and AGM. Similarly to the GGM and GGAM, it is possible relate the AGM to the AGAM. While the intuition behind this relation is similar to the generic case, it is a lot more complicated to formalize. The main complication stems from the fact that we do not have the useful formal limitations of the generic group model. Specifically, while it is easy to simply omit an oracle in an idealized model, formalizing the set of allowed operations of an arbitrary algorithm on some group or group action requires more rigor in order to make no arbitrary limitations. While this is possible via the abstraction of group schemes [2,4,26] (which can also be adapted to group actions), it is not very insightful.

Additionally, relating the AGAM to the AGM is not very useful. While we can adapt lower bounds from the GGM to the GGAM, we could only hope to adapt *upper bounds* from the AGAM to the AGM. However, it is very likely that there are actually better upper bounds that can be proven in the AGM as we have more freedom there. So we only get "upper bounds on upper bounds", which are of limited use.

Relation between AGAM and AGAM$^{\top}$. As in the GGAM, one generally can not directly translate results from the AGAM to the AGAM$^{\top}$ or vice versa. For the direction from AGAM to AGAM$^{\top}$, the examples from Section 4.2 apply as well. In the other direction, the situation is a bit more nuanced. Since the reduction can depend on the representations provided by the adversary, there are situations where the reduction only uses the twisting functionality of the group action, if the adversary does so as well. In this case the reduction can be moved from the AGAM$^{\top}$ to the AGAM. In all other cases, directly transfering reductions seems impossible and both models have to be considered.

5.2 Results in the Quantum Algebraic Group Action Model with Twists

As a warm-up, we show the equivalence of GA-DLOG and GA-CDH in the (Q)AGAM$^{\top}$. As the implication from GA-CDH to GA-DLOG is obvious, we only show the non-trivial implication.

Theorem 7 (GA-DLOG $\Rightarrow$ GA-CDH **in the (Q)AGAM$^{\top}$).** *Let $\mathcal{A}$ be an algebraic (quantum) $(\epsilon_{\mathcal{A}}, t_{\mathcal{A}})$-adversary against the* GA-CDH *problem, then there exists an $(\epsilon_{\mathcal{B}}, t_{\mathcal{B}})$-adversary $\mathcal{B}$ against the* GA-DLOG *problem with*

$$\epsilon_{\mathcal{A}} \leq \epsilon_{\mathcal{B}} \quad \textit{and} \quad t_{\mathcal{A}} \approx t_{\mathcal{B}} \ .$$

Formally, we prove the theorem in the $\mathrm{AGAM}^{\top}$. However, there are no quantum oracles and the proof adapts to the quantum setting without change.

Proof. We construct a reduction $\mathcal{B}$ against $\mathsf{GA\text{-}DLOG}$ as follows. $\mathcal{B}$ gets as input $x_1 := \mathfrak{a} \diamond \tilde{x}$ and chooses $\mathfrak{r} \xleftarrow{\$} \mathbb{Z}_N$. It computes $x_2 := \mathfrak{r} \diamond x_1$ and then runs the algebraic adversary $\mathcal{A}$ against $\mathsf{GA\text{-}CDH}$ on (x_1, x_2). At some point $\mathcal{A}$ will output a solution z as well as a representation $(i, \mathfrak{s}, b)$. Note that if $\mathcal{A}$ wins, then $z = (2\mathfrak{a} + \mathfrak{r}) \diamond \tilde{x}$. For each $i \in \{0, 1, 2\}$ and $b \in \{0, 1\}$, we show how we can solve the $\mathsf{GA\text{-}DLOG}$ challenge from $\mathfrak{s}$.

- Case 1 ($i = 0$): Since the twist maps the origin $\tilde{x}$ to itself, we get $\mathfrak{s} \diamond \tilde{x} = (2\mathfrak{a} + \mathfrak{r}) \diamond \tilde{x}$ for both $b \in \{0, 1\}$. Thus, we have to solve $2\mathfrak{a} = \mathfrak{s} - \mathfrak{r} \bmod N$. If $\gcd(2, N) = 2$ and 2 divides $\mathfrak{s} - \mathfrak{r}$, then we get two solutions and we test which is the correct one. Otherwise, there exists exactly one solution.
- Case 2 ($i = 1$): For $b = 0$, we get $\mathfrak{s} \diamond (\mathfrak{a} \diamond \tilde{x}) = (2\mathfrak{a} + \mathfrak{r}) \diamond \tilde{x}$. Thus, we get $\mathfrak{a} = \mathfrak{s} - \mathfrak{r} \bmod N$. For $b = 1$, we get $\mathfrak{s} \diamond (-\mathfrak{a} \diamond \tilde{x}) = (2\mathfrak{a} + \mathfrak{r}) \diamond \tilde{x}$ and we have to solve $3\mathfrak{a} = \mathfrak{s} - \mathfrak{r}$. If $\gcd(3, N) = 3$ and 3 divides $\mathfrak{s} - \mathfrak{r}$, then we get three solutions and we test which is the correct one. Otherwise, there exists exactly one solution.
- Case 3 ($i = 2$): For $b = 0$, we get $\mathfrak{s} \diamond ((\mathfrak{a} + \mathfrak{r}) \diamond \tilde{x}) = (2\mathfrak{a} + \mathfrak{r}) \diamond \tilde{x}$. Thus, $\mathfrak{a} = \mathfrak{s}$ is exactly the $\mathsf{GA\text{-}DLOG}$ solution. For $b = 1$, we get $\mathfrak{s} \diamond ((-\mathfrak{a} - \mathfrak{r}) \diamond \tilde{x}) = (2\mathfrak{a} + \mathfrak{r}) \diamond \tilde{x}$ and we have to solve $3\mathfrak{a} = \mathfrak{s} - 2\mathfrak{r}$, which we can do similar to case 2.

This concludes the proof of Theorem 7. □

An interesting new problem and its necessity to prove IND-CCA security of (plain) hashed ElGamal was put forward by [20]. They define different variants of the strong CDH problem, some of them allowing quantum queries to the decision oracle. We first recall the definition of the strongest version of their problem.

Definition 11 (Group Action Quantum Strong Computational Diffie-Hellman). *Let* $\mathsf{CEGA(T)} = (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ *be a cyclic effective group action (with twists). We say that an adversary* $\mathcal{A}$ *solves the group action quantum strong computational Diffie-Hellman problem* ($\mathsf{GA\text{-}QSt\text{-}CDH}$) *if* $\mathcal{A}^{\mathrm{O}}(\tilde{x}, \mathfrak{a} \diamond \tilde{x}, \mathfrak{b} \diamond \tilde{x}) = (a + b) \diamond \tilde{x}$ *for* $\mathfrak{a}, \mathfrak{b} \xleftarrow{\$} \mathbb{Z}_N$*, where* $\mathcal{A}$ *has access to decision oracles* $\mathrm{O} := \{\mathsf{GA\text{-}DDH}_{\mathfrak{a}}(|\cdot, \cdot\rangle), \mathsf{GA\text{-}DDH}_{\mathfrak{b}}(|\cdot, \cdot\rangle)\}$.

On basis-state inputs (y, z), $\mathsf{GA\text{-}DDH}_{\mathfrak{a}}$ *returns 1 if* $\mathfrak{a} \diamond y = z$ *and 0 otherwise.* $\mathsf{GA\text{-}DDH}_{\mathfrak{b}}$ *is defined equivalently. Note that superposition queries are then implicitly defined by linearity (i.e.,* $\mathrm{O}(\sum_x \alpha_x x) = \sum_x \alpha_x \mathrm{O}(x)$*).*

Remark 9. The $\mathsf{GA\text{-}QSt\text{-}CDH}$ assumption is called Double-Sided Fully Quantum Group Action Strong Computational Diffie-Hellman Problem in [20]. They define additional variants where the adversary can only access one of the two decision oracles or has only partial quantum access (i.e. one of the inputs is implicitly measured). The $\mathsf{GA\text{-}QSt\text{-}CDH}$ assumption is the strongest of these assumptions, so the result we show in Theorem 8 applies to the weaker assumptions as well.

Now we show that in the $\mathrm{QAGAM}^{\top}$, this problem can actually be reduced to $\mathsf{GA\text{-}DLOG}$ using the semi-classical oneway-to-hiding lemma (see Theorem 1).

Theorem 8 ($\mathsf{GA\text{-}DLOG} \Rightarrow \mathsf{GA\text{-}QSt\text{-}CDH}$ in the $\mathrm{QAGAM}^{\top}$). *Let* $\mathcal{A}$ *be an algebraic quantum* $(\epsilon_{\mathcal{A}}, t_{\mathcal{A}})$*-adversary against the* $\mathsf{GA\text{-}QSt\text{-}CDH}$ *problem making at most q decision oracle queries, then there exist* $(\epsilon_{\mathcal{B}}, t_{\mathcal{B}})$*-adversary* $\mathcal{B}$ *and* $(\epsilon_{\mathcal{C}}, t_{\mathcal{C}})$*-adversary* $\mathcal{C}$ *against* $\mathsf{GA\text{-}DLOG}$ *with*

$$\epsilon_{\mathcal{A}} \leq \sqrt{(q+1) \cdot \epsilon_{\mathcal{B}}} + \epsilon_{\mathcal{C}} \quad \textit{and} \quad t_{\mathcal{A}} \approx t_{\mathcal{B}} \approx t_{\mathcal{C}} \ .$$

Proof. The main difficulty of the proof is to show that the reduction can simulate the quantum $\mathsf{GA\text{-}DDH}$ oracles. The main observation then is that if the algebraic adversary queries $\mathsf{GA\text{-}DDH}$ on points where it might learn something interesting, it already had to solve $\mathsf{GA\text{-}DLOG}$. Therefore, we can assume that the adversary does not query those points (with noticeable probability amplitude) and we can remove them from the oracle using the semi-classical O2H lemma. This is done in the gamehop G_0 to G_1. On the remaining points where the adversary does not learn anything of interest, the reduction knows how to simulate $\mathsf{GA\text{-}DDH}$ perfectly and we can solve $\mathsf{GA\text{-}DLOG}$ from the algebraic $\mathsf{GA\text{-}CDH}$ solution, which bounds the probability of winning in G_1. We proceed with the formal proof.

Let $\mathcal{A}$ be a quantum algebraic adversary against the $\mathsf{GA\text{-}QSt\text{-}CDH}$ game. Consider the games given in Figure 2.

```
$G_0, G_1$
00 $\mathfrak{a}, \mathfrak{b}, \mathfrak{r} \xleftarrow{\$} \mathbb{Z}_N$
01 $x_1 := \mathfrak{a} \diamond \tilde{x}$
02 $x_2 := \mathfrak{b} \diamond \tilde{x}$                    $\backslash\backslash G_0$
03 $x_2 := \mathfrak{r} \diamond x_1$                    $\backslash\backslash G_1$
04 $z_{(i,\mathfrak{c},b)} \leftarrow \mathcal{A}^{O_\mathfrak{a}, O_\mathfrak{b}}(x_1, x_2)$
05 return $[\![z = (\mathfrak{a} + \mathfrak{b}) \diamond \tilde{x}]\!]$

Oracle $O_\mathfrak{a}\left(\left|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\right\rangle\right)$
06 $(i_\mathfrak{b}, i_\mathfrak{a}) := (\mathsf{msb}(i), \mathsf{lsb}(i))$                    $\backslash\backslash G_1$
07 $(j_\mathfrak{b}, j_\mathfrak{a}) := (\mathsf{msb}(j), \mathsf{lsb}(j))$
08 if $d \neq 0$
09     return 0
10 $\mathsf{res} := [\![((-1)^{b_2} j_\mathfrak{b} - (-1)^{b_1} j_\mathfrak{a})\mathfrak{r} + \mathfrak{d} - \mathfrak{c} = 0]\!]$
11 return res
12 return $|[\![\mathfrak{a} \diamond y = z]\!]\rangle$
```

Fig. 2. Games G_0-G_1 for the proof of Theorem 8. The oracle $O_\mathfrak{b}$ is simulated in the same way as $O_\mathfrak{a}$ except for an additional $\mathfrak{r}$ in the boolean test in line 10. The variable d is defined as in eq. (8). Lines 06 and 07 interpret i and j as a two digit binary number. Note that $i, j \leq 2$.

Game G_0. This is the definition of the GA-QSt-CDH game where we write $O_\mathfrak{a}$ for $\mathsf{GA\text{-}DDH}_\mathfrak{a}$ and likewise for $O_\mathfrak{b}$. We have

$$\Pr[G_0 \Rightarrow 1] = \epsilon_\mathcal{A}\,.$$

Game G_1. Here, we make two changes. The first change is a simple conceptual change. We use the random self-reducibility of GA-DLOG to set x_2 to $\mathfrak{r} \diamond x_1$ instead of $\mathfrak{b} \diamond \tilde{x}$ for $\mathfrak{r} \xleftarrow{\$} \mathbb{Z}_N$. The second change is in the simulation of the decision oracles. We reprogram certain points of the decision oracle $O_\mathfrak{a}$ and $O_\mathfrak{b}$ to always output 0. More specifically, we reprogram those points to 0 which would allow us to solve GA-DLOG if the adversary gave us elements where GA-DDH returns 1. By the semi-classical oneway-to-hiding lemma (Theorem 1) we can bound the difference between the games G_0 and G_1 by the probability of the adversary finding an element from the reprogrammed set of points $\mathcal{S}$. That is

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \sqrt{(q+1)\Pr\left[\textsc{Find} \mid \mathcal{A}^{G\setminus\mathcal{S}}\right]}\,,$$

where we define $\mathcal{S}$ as the set where the function SetTest defined in Figure 3 returns 1.

We bound the right-hand-side, showing that

$$\Pr\left[\textsc{Find} \,\middle|\, \mathcal{A}^{G\setminus\mathcal{S}}\right] \leq \epsilon_\mathcal{B}\,, \tag{6}$$

in the reduction $\mathcal{B}$ described in Figure 3. The reduction simulates the oracles as in G_1 and simulates the semi-classical O2H oracle by considering the cases described below.

Essentially, there are three cases when the algebraic adversary queries the decision oracle $O_\mathfrak{a}$. The first possibility is that the explanation of the adversary leads to an equation which we could solve for $\mathfrak{a}$ if GA-DDH returns 1 on that input. In that case the reduction solves GA-DLOG. If $\mathfrak{a}$ vanishes in the expression, we can simply simulate the GA-DDH oracle, since all variables necessary are known to the reduction. In the third case, $\mathfrak{a}$ does not vanish from the expression, but there is no solution, in which case 0 would have been returned anyway. To give a simple example of the first case, assume that $y = \mathfrak{c} \diamond x_1$ and $z = \mathfrak{d} \diamond \tilde{x}$, then $\mathfrak{a} = \mathfrak{d} - \mathfrak{c}$. To cover all cases we use variables $i_\mathfrak{a}$, $i_\mathfrak{b}$, $j_\mathfrak{a}$ and $j_\mathfrak{b}$ as defined in Figure 2 and can then derive the DDH expression

$$\mathfrak{a} + \underbrace{\mathfrak{c} + (-1)^{b_1}(i_\mathfrak{a}\mathfrak{a} + i_\mathfrak{b}(\mathfrak{r} + \mathfrak{a}))}_{\diamond\tilde{x}=y} \equiv \underbrace{\mathfrak{d} + (-1)^{b_2}(j_\mathfrak{a}\mathfrak{a} + j_\mathfrak{b}(\mathfrak{r} + \mathfrak{a}))}_{\diamond\tilde{x}=z} \mod N\,. \tag{7}$$

This equation can be rearranged to

$$\underbrace{(1 + (-1)^{b_1}(i_\mathfrak{a} + i_\mathfrak{b}) - (-1)^{b_2}(j_\mathfrak{a} + j_\mathfrak{b}))}_{=:d\in[-1,3]}\mathfrak{a} \equiv ((-1)^{b_2} j_\mathfrak{b} - (-1)^{b_1} j_\mathfrak{a})\mathfrak{r} + \mathfrak{d} - \mathfrak{c} \mod N \tag{8}$$

which we can use to either solve GA-DLOG when $\mathfrak{a}$ does not vanish and otherwise simulate $O_\mathfrak{a}$. Clearly, the variable $\mathfrak{a}$ vanishes in the expression if $d = 0$, enabling us to extract a solution if $d \neq 0$. This is tested in G_1 before using the expression to simulate $O_\mathfrak{a}$. However, in the case $d \neq 0$ we still have to consider

```
Reduction $\mathcal{B}(x_1 := \mathfrak{a} \diamond \tilde{x})$
00 $\mathfrak{r} \xleftarrow{\$} \mathbb{Z}_N$
01 $x_2 := \mathfrak{r} \diamond x_1$
02 $\mathcal{T} \leftarrow$ Run $\mathcal{A}^{O \setminus S}(x_1, x_2)$ until FIND
       and measure query register inputs
03 return FindSolution$(\mathcal{T}, x_1, x_2)$

Oracle $\mathrm{O}_S^{\mathsf{SC}}(|\psi_{in}, 0\rangle)$
04 Parse $\psi_{in}$ as $\left|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\right\rangle$
05 $b := 0$
06 $b \leftarrow$ Measure $\left[\mathsf{SetTest}\left(\left|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\right\rangle\right)\right]$
07 return $(|\psi'_{in}\rangle, b)$

SetTest $\left(\left|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\right\rangle\right)$
08 if $d \neq 0$
09     $\mathcal{T} := \mathsf{Solve}(\mathfrak{a}+\mathfrak{c}+(-1)^{b_1}(i_\mathfrak{a}\mathfrak{a}+i_\mathfrak{b}(\mathfrak{r}+\mathfrak{a})) =$
           $\mathfrak{d}+(-1)^{b_2}(j_\mathfrak{a}\mathfrak{a}+j_\mathfrak{b}(\mathfrak{r}+\mathfrak{a})))$
10     return $\exists \mathfrak{a}' \in \mathcal{T}\colon [\![\mathfrak{a}' \diamond \tilde{x} = x_1]\!]$
11 return 0

Oracle $\mathrm{O}_\mathfrak{a} \setminus \mathcal{S}(|\psi_{in}, \psi_{out}\rangle)$
12 $|\psi'_{in}, b\rangle \leftarrow \mathrm{O}_S^{\mathsf{SC}}(|\psi_{in}, 0\rangle)$
13 if $b = 1$
14     FIND := 1
15 return $U_O(|\psi'_{in}, \psi_{out}\rangle)$

FindSolution$(\mathcal{T}, x_1, x_2)$
16 Parse $\mathcal{T}$ as $(y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)})$
17 Run SetTest$(y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)})$
   returning $\mathfrak{a}'$ s.t. $\mathfrak{a}' \diamond \tilde{x} = x_1$
18 return $\mathfrak{a}'$
```

Fig. 3. Reduction $\mathcal{B}$ for bounding the difference between G_0 and G_1 for the proof of Theorem 8. The function Solve solves for $\mathfrak{a}$ and outputs a set of (possibly multiple) solutions. The oracle $\mathrm{O}_\mathfrak{a}$ is simulated as in G_1. The simulation of $\mathrm{O}_\mathfrak{b}$ is similar and explained in the proof. Line 17 is abuse of notation and queries a search version of SetTest instead of the decisional version. The variables $i_\mathfrak{a}$, $i_\mathfrak{b}$, $j_\mathfrak{a}$ and $j_\mathfrak{b}$ are defined as in Figure 2. The variable d is defined as in eq. (8).

the fact that $d \mid N$ but d does not divide the right hand side of eq. (8). Referring back to Lemma 2 we therefore have no solutions for $\mathfrak{a}$. This is a non-issue as we can just return 0 in that case as well. If $d = 0$ we test whether the right hand side evaluates to 0 as well. For simulating $\mathrm{O}_\mathfrak{b}(y, z) = \mathrm{O}_{\mathfrak{r}+\mathfrak{a}}(y, z) = \mathrm{O}_\mathfrak{a}(\mathfrak{r} \diamond y, z)$ we proceed in the same way except that $\mathfrak{r}$ is added to the left side of eq. (7).

We have just shown Equation 6. It remains to reduce G_1 to GA-DLOG. That is,

$$\Pr[\mathrm{G}_1 \Rightarrow 1] \leq \epsilon_\mathcal{C} \,.$$

This is straightforward, the reduction $\mathcal{C}$ gets the GA-DLOG challenge $x_1 = \mathfrak{a} \diamond \tilde{x}$, samples $\mathfrak{r} \xleftarrow{\$} \mathbb{Z}_N$ and sets $x_2 = \mathfrak{r} \diamond x_1$. Then $\mathcal{C}$ simulates G_1 and receives the GA-CDH solution and solves GA-DLOG as in Theorem 7. Adding up the terms yields the claimed bound and concludes the proof. □

Now we want to analyze the strong square inverse Diffie-Hellman assumption introduced in [1] to prove security of their password-authenticated key exchange protocol. As pointed out in [20], a security proof in the quantum random oracle model will rely on a stronger version of the assumption, where decision oracles are queried in quantum superposition. This is similar to the case of hashed ElGamal and the GA-QSt-CDH assumption.

Definition 12 (Group Action Quantum Strong Square-Inverse Diffie-Hellman). *Let* CEGA(T) $= (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ *be a cyclic effective group action (with twists). We say that an adversary $\mathcal{A}$ solves the group action quantum strong square-inverse Diffie-Hellman problem* (GA-QSt-SqInvDH) *if* $\mathcal{A}^{\mathrm{O}}(\tilde{x}, \mathfrak{a} \diamond \tilde{x}) = (y, 2\mathfrak{a} \diamond y, -\mathfrak{a} \diamond y)$ *for* $\mathfrak{a} \xleftarrow{\$} \mathbb{Z}_N$ *and some* $y \in \mathcal{X}$. *Here $\mathcal{A}$ has access to decision oracles* $\mathrm{O} := \{\mathsf{GA\text{-}DDH}_\mathfrak{a}(|\cdot\rangle, |\cdot\rangle), \mathsf{GA\text{-}DDH}_{2\mathfrak{a}}(|\cdot\rangle, |\cdot\rangle)\}$ *which are defined similarly to those in Definition 11.*

Theorem 9. *Let $\mathcal{A}$ be an $(\epsilon_\mathcal{A}, t_\mathcal{A})$-algebraic quantum adversary against the* GA-QSt-SqInvDH *problem that issues at most q decision oracle queries, then there exist an $(\epsilon_\mathcal{B}, t_\mathcal{B})$-adversary $\mathcal{B}$ and an $(\epsilon_\mathcal{C}, t_\mathcal{C})$-adversary $\mathcal{C}$ against the* GA-DLOG *problem with*

$$\epsilon_\mathcal{A} \leq \sqrt{(q+1) \cdot \epsilon_\mathcal{B}} + \epsilon_\mathcal{C} \quad \textit{and} \quad t_\mathcal{A} \approx t_\mathcal{B} \approx t_\mathcal{C} \,.$$

Here the square-root term comes from the fact that we allow quantum queries to the decision oracles. When allowing for classical queries only, the bound would be tight. The proof is very similar to that of Theorem 8 and we defer it to Appendix C.

```
Game IND-CCA1($\mathcal{A}$)
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
    $b' \leftarrow \mathcal{A}^{\textsc{Enc},\textsc{Decaps}}(\mathsf{pk})$
    return $[\![b = b']\!]$

Oracle Enc                                  \\ only once
    $b \xleftarrow{\$} \{0,1\}$
    $(\mathsf{ct}^*, K_0) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
    $K_1 \xleftarrow{\$} \mathcal{K}$
    return $(\mathsf{ct}^*, K_b)$

Oracle Decaps(ct)                           \\ only before Enc
    return $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
```

Fig. 4. The IND-CCA1 game for a key encapsulation mechanism KEM.

```
Gen
    $\mathsf{sk} := \mathfrak{a} \xleftarrow{\$} \mathbb{Z}_N$
    $\mathsf{pk} := \mathfrak{a} \diamond \tilde{x}$
    return $(\mathsf{pk}, \mathsf{sk})$

Encaps(pk)
    $\mathfrak{b} \xleftarrow{\$} \mathbb{Z}_N$
    $\mathsf{ct} := \mathfrak{b} \diamond \tilde{x}$
    $K := \mathfrak{b} \diamond \mathsf{pk}$
    return $(\mathsf{ct}, K)$

Decaps(sk, ct)
    $K := \mathsf{sk} \diamond \mathsf{ct}$
    return $K$
```

Fig. 5. The ElGamal KEM for $\mathsf{CEGA(T)} = (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$.

5.3 Security Analysis of ElGamal in the Quantum Algebraic Group Action Model with Twists

Similar to the analysis in the AGM of [23], we can prove IND-CCA1 security of group action ElGamal. Going even one step further, we are also able to analyze IND-QCCA1 security in the QAGAM$^\top$. This security notion, where the adversary is allowed to ask for decapsulation queries in quantum superposition, was first proposed by Boneh and Zhandry in [13]. We recall the definition of a key encapsulation mechanism and the group action q-decisional Diffie-Hellman assumption below. This assumption is similar to those in Definitions 8 and 9. Lastly, we define the ElGamal KEM in Figure 5.

KEY ENCAPSULATION MECHANISM. Let $\mathcal{PK}$, $\mathcal{SK}$, $\mathcal{C}$, $\mathcal{K}$ be sets. A *key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of the following three algorithms:
- Gen: The key generation algorithm outputs a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.
- Encaps(pk): On input a public key pk, the encapsulation algorithm returns a ciphertext $\mathsf{ct} \in \mathcal{C}$ and a key $K \in \mathcal{K}$, where ct is an encapsulation of K.
- Decaps(sk, ct): On input a secret key sk and a ciphertext ct, the decapsulation algorithm returns a key $K \in \mathcal{K}$ or a special failure symbol $\perp$.

We require perfect correctness, i.e. for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$, $(\mathsf{ct}, K) \leftarrow \mathsf{Encaps}(\mathsf{pk})$, we have $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}) = K$.

IND-CCA1 SECURITY. We define the IND-CCA1 security game (aka. lunchtime security) in Figure 4. We say that an adversary $\mathcal{A}$ solves the IND-CCA1 game if the game outputs 1. Analogously, we define the IND-QCCA1 security game, where the only difference from the regular IND-CCA1 game is that an adversary has *quantum* access to the decapsulation oracle.

Definition 13 (Group Action q-Decisional Diffie-Hellman). *Let* $\mathsf{CEGA(T)} = (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$. *We say that an adversary $\mathcal{A}$ solves the group action q-decisional Diffie-Hellman* (GA-q-DDH) *if*

$$\mathcal{A}(\tilde{x}, \mathfrak{a} \diamond \tilde{x}, 2\mathfrak{a} \diamond \tilde{x}, \ldots, q\mathfrak{a} \diamond \tilde{x}, \mathfrak{b} \diamond \tilde{x}, z_r) = r \ ,$$

where $\mathfrak{a}, \mathfrak{b} \xleftarrow{\$} \mathbb{Z}_N$, $z_0 := (\mathfrak{a} + \mathfrak{b}) \diamond \tilde{x}$, $z_1 \xleftarrow{\$} \mathcal{X}$ *and* $r \xleftarrow{\$} \{0,1\}$.

Theorem 10 (GA-q-DDH $\implies$ IND-QCCA1 of ElGamal). *Let $\mathcal{A}$ be an algebraic quantum $(\epsilon_\mathcal{A}, t_\mathcal{A})$-adversary against the* IND-QCCA1 *security of ElGamal KEM making at most $q-1$ quantum decapsulation queries, then there exists a quantum $(\epsilon_\mathcal{B}, t_\mathcal{B})$-adversary $\mathcal{B}$ against* GA-q-DDH *with*

$$\epsilon_\mathcal{A} \leq \epsilon_\mathcal{B} \quad \text{and} \quad t_\mathcal{A} \approx t_\mathcal{B} \ .$$

Proof. We describe how the reduction works classically. Simulation of the quantum decapsulation oracle follows from linearity of quantum states and from the fact that all computations are deterministic from their inputs.

Let $(x_1, x_2, \ldots, x_q, y, z) := (\mathfrak{a} \diamond \tilde{x}, 2\mathfrak{a} \diamond \tilde{x}, \ldots, q\mathfrak{a} \diamond \tilde{x}, \mathfrak{b} \diamond \tilde{x}, z)$ be the GA-q-DDH challenge. The main idea of the proof is to use the x_i to simulate the decapsulation oracle together with the algebraic explanation provided by the algebraic adversary. If the adversary $\mathcal{A}$ uses twists, the reduction $\mathcal{B}$ can still simulate decapsulation queries by twisting the appropriate x_i. We will therefore define $x_{-i} := x_i^t$. The main observation is that an adversary can learn higher "powers" of $\mathfrak{a} \diamond \tilde{x}$, i.e. $k\mathfrak{a} \diamond \tilde{x}$ for $k \in [q]$, by querying $\mathfrak{a} \diamond \tilde{x}$ to the decapsulation oracle. $\mathcal{A}$ can then query the decapsulation oracle on ciphertexts which depend on answers to previous decapsulations and so on. Then the i-th answer to the decapsulation oracle can be written as

$$K_i = \mathfrak{d}_i \diamond x_{j_i} = (j_i\mathfrak{a} + \mathfrak{d}_i) \diamond \tilde{x}$$

where $|j_i| \leq i$. For the $i+1$-th decapsulation query, we then have

$$\begin{aligned}\mathsf{Decaps}_{\mathfrak{a}}(c_{(k,\mathfrak{d},r)}) &= \mathfrak{a} \diamond c = (a + \mathfrak{d}) \diamond K_k^r \\ &= (\mathfrak{a} + (-1)^t i_k \mathfrak{a} + \mathfrak{d}_k + \mathfrak{d}) \diamond \tilde{x} = (\mathfrak{d} + \mathfrak{d}_k) \diamond x_{(-1)^t i_k + 1},\end{aligned}$$

where $K_k^r = K_k$ if $b = 0$ and K_k^t otherwise. With $i_{j+1} := (-1)^t i_k + 1$ and $\mathfrak{d}_i := \mathfrak{d} + \mathfrak{d}_k$, we get a representation for K_{j+1} as above, which can in turn be used to simulate the next decapsulation. Since $\mathcal{A}$ makes at most $q-1$ decapsulation queries, we always have $|i_k| \leq q$, so the simulation always works.

Being able to simulate decapsulation queries, the adversary sets the ElGamal randomness carrier to be $\mathfrak{b} \diamond \tilde{x}$ and sets the KEM key to be z and uses the distinguishing bit to decide if it is in the real or random case of the GA-q-DDH problem. □

Acknowledgments. The work of Julien Duman was supported by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM Research Hub under Grant 16KISK037. Dominik Hartmann was supported by the European Union (ERC AdG REWORC - 101054911). Eike Kiltz was supported by the Deutsche Forschungsgemeinschaft (DFG, German research Foundation) as part of the Excellence Strategy of the German Federal and State Governments – EXC 2092 CASA - 390781972, and by the European Union (ERC AdG REWORC - 101054911). Sabrina Kunzweiler, Jonas Lehmann and Doreen Riepel were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

1. Abdalla, M., Eisenhofer, T., Kiltz, E., Kunzweiler, S., Riepel, D.: Password-authenticated key exchange from group actions. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. pp. 699–728. Springer Nature Switzerland, Cham (2022)
2. Agrikola, T., Hofheinz, D.: Interactively secure groups from obfuscation. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 341–370. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_12
3. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
4. Albrecht, M.R., Farshim, P., Hofheinz, D., Larraia, E., Paterson, K.G.: Multilinear maps from obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 446–473. Springer, Heidelberg (Jan 2016). https://doi.org/10.1007/978-3-662-49096-9_19
5. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 269–295. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_10
6. Baghery, K., Cozzo, D., Pedersen, R.: An isogeny-based ID protocol using structured public keys. In: Paterson, M.B. (ed.) 18th IMA International Conference on Cryptography and Coding. LNCS, vol. 13129, pp. 179–197. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92641-0_9
7. Bao, F., Deng, R.H., Zhu, H.: Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 03. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (Oct 2003)
8. Bauer, B., Fuchsbauer, G., Loss, J.: A classification of computational assumptions in the algebraic group model. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 121–151. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_5
9. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331 (2004), https://eprint.iacr.org/2004/331

10. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_9
11. Biasse, J.F., Song, F.: Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In: Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms. pp. 893–902. SIAM (2016)
12. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology **21**(2), 149–177 (Apr 2008). https://doi.org/10.1007/s00145-007-9005-7
13. Boneh, D., Zhandry, M.: Secure signatures and chosen ciphertext security in a quantum computing world. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 361–379. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_21
14. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Report 2022/975 (2022), https://eprint.iacr.org/2022/975
15. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
16. Cheon, J.H.: Discrete logarithm problems with auxiliary inputs. Journal of Cryptology **23**(3), 457–476 (Jul 2010). https://doi.org/10.1007/s00145-009-9047-0
17. Cohen, H., Lenstra, H.: Heuristics on class groups. In: Number theory, pp. 26–36. Springer (1984)
18. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
19. De Feo, L., Meyer, M.: Threshold schemes from isogeny assumptions. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 187–212. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45388-6_7
20. Duman, J., Hartmann, D., Kiltz, E., Kunzweiler, S., Lehmann, J., Riepel, D.: Group action key encapsulation and non-interactive key exchange in the qrom. ASIACRYPT 2022 (2022)
21. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D.: A thorough treatment of highly-efficient NTRU instantiations. Cryptology ePrint Archive, Report 2021/1352 (2021), https://eprint.iacr.org/2021/1352
22. Ettinger, M., Høyer, P.: On quantum algorithms for noncommutative hidden subgroups. Advances in Applied Mathematics **25**(3), 239–251 (2000). https://doi.org/https://doi.org/10.1006/aama.2000.0699, https://www.sciencedirect.com/science/article/pii/S0196885800906997
23. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2
24. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: 28th ACM STOC. pp. 212–219. ACM Press (May 1996). https://doi.org/10.1145/237814.237866
25. Hafner, J.L., McCurley, K.S.: A rigorous subexponential algorithm for computation of class groups. Journal of the American mathematical society **2**(4), 837–850 (1989)
26. Kastner, J., Pan, J.: Towards instantiating the algebraic group model. Cryptology ePrint Archive, Report 2019/1018 (2019), https://eprint.iacr.org/2019/1018
27. Katz, J., Zhang, C., Zhou, H.S.: An analysis of the algebraic group model. Cryptology ePrint Archive, Report 2022/210 (2022), https://eprint.iacr.org/2022/210
28. Kim, T.: Security analysis of group action inverse problem with auxiliary inputs with application to CSIDH parameters. In: Seo, J.H. (ed.) ICISC 19. LNCS, vol. 11975, pp. 165–174. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-40921-0_10
29. Kobayashi, H., Gall, F.: Dihedral hidden subgroup problem: A survey. Information and Media Technologies **1**, 178–185 (06 2006). https://doi.org/10.11185/imt.1.178
30. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 451–479. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0_18
31. Kuperberg, G.: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM Journal on Computing **35**(1), 170–188 (2005)
32. Maino, L., Martindale, C.: An attack on SIDH with arbitrary starting curve. Cryptology ePrint Archive, Report 2022/1026 (2022), https://eprint.iacr.org/2022/1026
33. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_2
34. Mizuide, T., Takayasu, A., Takagi, T.: Tight reductions for Diffie-Hellman variants in the algebraic group model. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 169–188. Springer, Heidelberg (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_9

35. Montgomery, H., Zhandry, M.: Full quantum equivalence of group action DLog and CDH, and more. ASIACRYPT 2022 (2022)
36. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_16
37. Pointcheval, D., Wang, G.: VTBPEKE: Verifier-based two-basis password exponential key exchange. In: Karri, R., Sinanoglu, O., Sadeghi, A.R., Yi, X. (eds.) ASIACCS 17. pp. 301–312. ACM Press (Apr 2017)
38. Robert, D.: Breaking SIDH in polynomial time. Cryptology ePrint Archive, Report 2022/1038 (2022), https://eprint.iacr.org/2022/1038
39. Rostovtsev, A., Stolbunov, A.: Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145 (2006), https://eprint.iacr.org/2006/145
40. Shanks, D.: Class number, a theory of factorization, and genera. In: Proc. of Symp. Math. Soc., 1971. vol. 20, pp. 41–440 (1971)
41. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700
42. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_18
43. Yoneyama, K.: Post-quantum variants of ISO/IEC standards: Compact chosen ciphertext secure key encapsulation mechanism from isogeny. In: Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop. p. 13–21. SSR'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338500.3360336
44. Zhandry, M.: Redeeming reset indifferentiability and applications to post-quantum security. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 518–548. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_18
45. Zhandry, M.: To label, or not to label (in generic groups). In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. pp. 66–96. Springer Nature Switzerland, Cham (2022)

A Generic Models for General Abelian Group Actions

In the models presented in our paper, we restricted our attention to group actions $(\mathcal{G}, \mathcal{X}, \star)$, where $\mathcal{G}$ is cyclic. This choice is based on the fact that for all known and cryptographically relevant instantiations, the group $\mathcal{G}$ is at least close to being cyclic, cf. Section 3.2. Since it is possible that future developments in cryptography might lead to new examples of group actions, we now explain how our definitions can be extended to general abelian group actions.

We start by defining the natural generalization of a CEGA (Definition 4). In essence, this is the same as a known-order effective group action introduced in [3].

Definition 14 (Abelian Known-Order Effective Group Action). *Let $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ be an effective group action satisfying the following properties:*

1. *The group $\mathcal{G}$ is finite of known rank $r \in \mathbb{N}$ and there exist known integers $N_1, \dots, N_r$ with $N_{i+1} \mid N_i$ for $i \in \{1, \dots, r-1\}$ such that*
$$\mathcal{G} \cong \mathbb{Z}/N_1\mathbb{Z} \times \cdots \times \mathbb{Z}/N_r\mathbb{Z}.$$
2. *There exists a generating set $(g_1, \dots, g_r) \in \mathcal{G}^r$ with $ord(g_i) = N_i$, i.e. $\mathcal{G} = \langle g_1, \dots, g_r \rangle$ with known representation.*
3. *For any element $h \in \mathcal{G}$, the elements $(\mathfrak{a}_1, \dots, \mathfrak{a}_r)$ with $\mathfrak{a}_i \in \mathbb{Z}/N_i\mathbb{Z}$ so that $h = g_1^{\mathfrak{a}_1} \circ \cdots \circ g_r^{\mathfrak{a}_r}$ are efficiently computable.*

Then we say that $(\mathcal{G}, \mathcal{X}, \star, \tilde{x}, (g_1, \dots, g_r))$ is an abelian known-order effective group action (KEGA).

In other words, an abelian known-order effective group action is an EGA *for which there exists an isomorphism $\phi : \mathcal{G} \to \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_r}$ efficiently computable in both directions. We therefore denote any* KEGA *equivalently by $(\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_r}, \mathcal{X}, \diamond, \tilde{x})$, where*

$$\begin{aligned} \diamond : (\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_r}) \times \mathcal{X} &\to \mathcal{X} \\ ((\mathfrak{a}_1, \dots, \mathfrak{a}_r), x) &\mapsto (g_1^{\mathfrak{a}_1} \circ \cdots \circ g_r^{\mathfrak{a}_r}) \star x. \end{aligned}$$

The order of the group is denoted by $N = \prod_{i=1}^{r} N_i$. To specify the orders of the cylic subgroups in the decomposition, we use the notation $N = (N_1, \dots, N_r)$.

Similar to the definition of CEGAs, we also define known-order group actions with twists.

Definition 15 (Abelian Known-Order Effective Group Action with Twists). *Let $(\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_r}, \mathcal{X}, \diamond, \tilde{x})$ be a* KEGA. *We call it Abelian Known-Order Group Action with Twists* (KEGAT) *if there is an efficient algorithm that, given $x = \mathfrak{a} \diamond \tilde{x}$, computes $x^t = -\mathfrak{a} \diamond \tilde{x}$.*

Generic Group Action Models. Essentially, the definitions of the GGAM and the GGAM$^\top$ do not change when replacing CEGA by a KEGA. In the following, we call the resulting models KEGA-GGAM and KEGA-GGAM$^\top$ respectively. Proofs in these models become technically more intricate. This is mostly due to the fact that group elements are now of the form $\mathfrak{a} = (\mathfrak{a}_1, \dots, \mathfrak{a}_r) \in \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_r}$. To illustrate the application of such models, we prove the following theorem which is the analogue of Theorem 2 in the GGAM$^\top$.

Theorem 11. *Let* KEGA-*GGAM*$^\top$ *be of order $N = (N_1, \dots, N_r)$. For every generic (ϵ, t, q)-algorithm $\mathcal{A}$ winning the* GA-DLOG *game we have*

$$\epsilon \leq \frac{2^{r^*} q^2}{N}, \quad \textit{where } r^* = \#\{i \mid \gcd(2, N_i) = 2\}.$$

Proof. The proof follows the same idea as the proof of Theorem 2. The main difference is that an indeterminate $\mathbf{X}$ now has to be understood as a vector, where the entry i is viewed as a variable over $\mathbb{Z}_{N_i}$. We denote $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_r)$ and a polynomial $f(\mathbf{X})$ is defined as system of polynomials. In other words:

$$f(\mathbf{X}) = \left\{ \begin{array}{c} f_1(\mathbf{X}_1) \in \mathbb{Z}_{N_1}[\mathbf{X}_1] \\ \vdots \\ f_r(\mathbf{X}_r) \in \mathbb{Z}_{N_r}[\mathbf{X}_r] \end{array} \right\}.$$

We need to bound the probability that an adversary makes two queries $f(\mathbf{X})$ and $g(\mathbf{X})$ that result in different labels but both polynomials evaluate to the same value. Note that two polynomials $f(\mathbf{X})$ and $g(\mathbf{X})$ evaluate to the same value if and only if $f_i(\mathbf{X}_i)$ and $g_i(\mathbf{X}_i)$ evaluate to the same value for all $i \in \{1, \dots, r\}$.

An adversary only has access to $\mathcal{O}^{\mathsf{exp}}$ and $\mathcal{O}^{\mathsf{tw}}$, hence the differences of polynomials it can compute are of the form

$$\mathfrak{g} \pm \mathbf{X} = \left\{ \begin{array}{c} \mathfrak{g}_1 \pm \mathbf{X}_1 \in \mathbb{Z}_{N_1}[\mathbf{X}_1] \\ \vdots \\ \mathfrak{g}_r \pm \mathbf{X}_r \in \mathbb{Z}_{N_r}[\mathbf{X}_r] \end{array} \right\}, \quad \mathfrak{g} \pm 2\mathbf{X} = \left\{ \begin{array}{c} \mathfrak{g}_1 \pm 2\mathbf{X}_1 \in \mathbb{Z}_{N_1}[\mathbf{X}_1] \\ \vdots \\ \mathfrak{g}_r \pm 2\mathbf{X}_r \in \mathbb{Z}_{N_r}[\mathbf{X}_r] \end{array} \right\}.$$

Just as in the proof of Theorem 2 an equation $\mathfrak{g}_i \pm \mathbf{X}_i = 0$ has precisely one solution, and an equation $\mathfrak{g}_i \pm 2\mathbf{X}_i = 0$ has at most two solutions if $2 \mid N$ and one solution otherwise. In conclusion, the equation $\mathfrak{g} \pm \mathbf{X} = 0$ has precisely one solution. And the equation $\mathfrak{g} \pm 2\mathbf{X} = 0$ has at most 2^{r^*} solutions, where $r^* = \#\{i \mid \gcd(N_i, 2) = 2\}$. The overall bound follows. □

In a similar way, it is also possible to translate the proof of Theorem 3 to the KEGA-GGAM$^\top$ which results in the following theorem.

Theorem 12. *For every generic (ϵ, t, q)-algorithm $\mathcal{A}$ winning* GA-k-PDDH *in the* KEGA-*GGAM*$^\top$ *of order $N = (N_1, \dots, N_r)$ we have*

$$\epsilon \leq \frac{1}{2} + \frac{2 \cdot d_{max} \cdot q^2}{N},$$

where $d_{max} = \prod_{i=1}^{r} d_{i,max}$ and $d_{i,max} = \max\{\gcd(2k, N_i), \gcd(k \pm 1, N_i)\}$ for $i \in \{1, \dots, r\}$.

Proof. The proof follows the same idea as the proof of Theorem 3 for generic adversaries in the GGAM$^\top$. The adaptions to KEGA-GGAM$^\top$ are similar to those in the proof of Theorem 11. □

B Ettinger-Høyer Algorithm

In [22] the authors reduce the DHSP to the following computational problem which we call the *Trigonometric Approximation Problem.*

Definition 16 (Trigonometric Approximation Problem). *Let $M, m \in \mathbb{N}$ be public parameters. For some uniformly random $k_0 \xleftarrow{\$} \mathbb{N}$ with $k_0 < M$ denote by Z the discrete random variable defined by the probability mass function*

$$\Pr[Z = z] = \alpha \cos^2(\pi k_0 z / M) \qquad (0 \leq z < M)$$

where $\alpha = 1/M$ if $k_0 = 0$ or $k_0 = M/2$ and $\alpha = 2/M$ otherwise. On input m independent random samples $z_1, \dots, z_m$ from Z, the Trigonometric Approximation Problem (TAP) *requires to compute the secret k_0.*

Here the parameters of TAP relate to GA-DLOG in the following way: For a fixed CEGA $= (\mathbb{Z}_N, \mathcal{X}, \diamond, \tilde{x})$ the parameter M is equivalent to the order N, m is equal to the number of queries q to $\mathcal{O}^{\mathsf{exp}}$ and k_0 corresponds to the secret exponent $\mathfrak{a}$. Specifically, k_0 yields the solution to an instance of the DHSP problem, which in turn can be used to solve the GA-DLOG instance $\mathfrak{a} \diamond \tilde{x}$.

Ettinger and Høyer solve TAP for $m \geq \lceil 64 \ln M \rceil$ by observing that any $\kappa \in \{1, \dots, \lfloor M/2 \rfloor\}$ that maximizes the sum

$$\sum_{i=1}^{m} \cos(2\pi\kappa z_i / M)$$

is equal to k_0 with probability at least $1 - \frac{1}{2M}$. The corresponding value of κ is found via a Grover search [24]. Therefore the Ettinger-Høyer algorithm is a (ϵ, t, q)-algorithm against GA-DLOG with $\epsilon = 1 - \frac{1}{2N}$, $t \in \mathcal{O}(\sqrt{N})$ and $q = \lceil 64 \ln N \rceil$, striking a specific trade-off between t and q that is asymptotically optimal with respect to the number of oracle queries.

```
$G_0, G_1$
$\mathfrak{a} \xleftarrow{\$} \mathbb{Z}_N$
$x := \mathfrak{a} \diamond \tilde{x}$
$(y_0, y_1, y_2) \leftarrow \mathcal{A}^{O_\mathfrak{a}, O_{2\mathfrak{a}}}(\tilde{x}, x)$
return $[\![ y_1 = 2\mathfrak{a} \diamond y_0 \wedge y_2 = -\mathfrak{a} \diamond y_0 ]\!]$

Oracle $O_\mathfrak{a}\left(\left|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\right\rangle\right)$
if $(1 + (-1)^{b_1} i - (-1)^{b_2} j) \neq 0$        $\backslash\backslash G_1$
    return 0
return $[\![ \mathfrak{c} = \mathfrak{d} ]\!]$
return $|[\![ \mathfrak{a} \diamond y = z ]\!]\rangle$

Oracle $O_{2\mathfrak{a}}\left(\left|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\right\rangle\right)$
if $(2 + (-1)^{b_1} i - (-1)^{b_2} j) \neq 0$        $\backslash\backslash G_1$
    return 0
return $[\![ \mathfrak{c} = \mathfrak{d} ]\!]$
return $|[\![ 2\mathfrak{a} \diamond y = z ]\!]\rangle$
```

Fig. 6. Games G_0-G_1 for the proof of Theorem 9. Note that for the solution (y_0, y_1, y_2) the adversary also provides a representation $(\mathfrak{i}_0, \mathfrak{s}_0, b_0)$, $(\mathfrak{i}_1, \mathfrak{s}_1, b_1)$ and $(\mathfrak{i}_2, \mathfrak{s}_2, b_2)$, respectively.

C Proof of Theorem 9

This section is dedicated to the proof of Theorem 9. For the convenience of the reader, we first repeat the statement of the theorem.

Theorem 9. *Let $\mathcal{A}$ be an $(\epsilon_\mathcal{A}, t_\mathcal{A})$-algebraic quantum adversary against the* GA-QSt-SqInvDH *problem that issues at most q decision oracle queries, then there exist an $(\epsilon_\mathcal{B}, t_\mathcal{B})$-adversary $\mathcal{B}$ and an $(\epsilon_\mathcal{C}, t_\mathcal{C})$-adversary $\mathcal{C}$ against the* GA-DLOG *problem with*

$$\epsilon_\mathcal{A} \leq \sqrt{(q+1) \cdot \epsilon_\mathcal{B}} + \epsilon_\mathcal{C} \quad \text{and} \quad t_\mathcal{A} \approx t_\mathcal{B} \approx t_\mathcal{C} \ .$$

Proof. Let $\mathcal{A}$ be a quantum algebraic adversary against the GA-QSt-SqInvDH game. Consider the games given in Figure 6.

Game G_0. This is the definition of the GA-QSt-SqInvDH game, where we write $O_\mathfrak{a}$ and $O_{2\mathfrak{a}}$ for the corresponding decision oracles GA-DDH$_\mathfrak{a}$ and GA-DDH$_{2\mathfrak{a}}$. We have

$$\Pr[G_0 \Rightarrow 1] = \epsilon_\mathcal{A} \, .$$

Game G_1. In game G_1, we change the simulation of the decision oracles. We reprogram certain points of the decision oracle $O_\mathfrak{a}$ and $O_{2\mathfrak{a}}$ to always output 0. More specifically, we reprogram those points to 0 which would allow us to solve GA-DLOG if the adversary gave us elements where GA-DDH returns 1. By the semi-classical oneway-to-hiding lemma (Theorem 1) we can bound the difference between the games G_0 and G_1 by the probability of the adversary finding an element from the reprogrammed set of points $\mathcal{S}$. This means

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \sqrt{(q+1) \Pr\left[\text{FIND} \mid \mathcal{A}^{G \setminus \mathcal{S}}\right]} \, ,$$

where $\mathcal{S}$ is defined as the set where the function SetTest defined in Figure 7 returns 1.

We bound the right-hand-side, showing that

$$\Pr\left[\text{FIND} \,\middle|\, \mathcal{A}^{G \setminus \mathcal{S}}\right] \leq \epsilon_\mathcal{B} \, , \tag{9}$$

in the reduction $\mathcal{B}$ described in Figure 7. The reduction simulates the oracles as in G_1 and simulates the semi-classical O2H oracle by considering the cases described below.

The first oracle is identical to the GA-DDH$_\mathfrak{a}$ oracle of in the GA-QSt-CDH assumption. Specifically, $\mathcal{B}$ can either simulate the oracle correctly or already extract a solution $\mathfrak{a}'$ for its GA-DLOG challenge. To cover all cases, we derive one expression based on the representations the adversary provides. Recall that any input to $O_\mathfrak{a}$ will be of the form $\left|y_{(i,\mathfrak{c},b_1)} z_{(j,\mathfrak{d},b_2)}\right\rangle$. If the adversary queries a real DDH tuple, then the following holds:

$$\mathfrak{a} + \underbrace{\mathfrak{c} + (-1)^{b_1} i\mathfrak{a}}_{\diamond \tilde{x} = y} \equiv \underbrace{\mathfrak{d} + (-1)^{b_2} j\mathfrak{a}}_{\diamond \tilde{x} = z} \mod N \, ,$$

```
Reduction $\mathcal{B}(\tilde{x}, x := \mathfrak{a} \diamond \tilde{x})$
$\mathcal{T} \leftarrow$ Run $\mathcal{A}^{O \setminus S}(\tilde{x}, x)$ until FIND
    and measure query register inputs
return FindSolution$(\mathcal{T}, \tilde{x}, x)$

Oracle $\mathrm{O}_S^{\mathsf{SC}}(|\psi_{in}, 0\rangle)$
Parse $\psi_{in}$ as $|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\rangle$
$b := 0$
$b \leftarrow$ Measure $[\mathsf{SetTest}(|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\rangle)]$
return $(|\psi'_{in}\rangle, b)$

SetTest $(|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\rangle)$
$d := (1 + (-1)^{b_1} i - (-1)^{b_2} j)$
if $d \neq 0$
    $\mathcal{T} := \mathsf{Solve}(d\mathfrak{a} = \mathfrak{d} - \mathfrak{c})$
    return $\exists \mathfrak{a}' \in \mathcal{T} \colon [\![\mathfrak{a}' \diamond \tilde{x} = x_1]\!]$
return 0

Oracle $\mathrm{O}_\mathfrak{a} \setminus \mathcal{S}(|\psi_{in}, \psi_{out}\rangle)$
$|\psi'_{in}, b\rangle \leftarrow \mathrm{O}_S^{\mathsf{SC}}(|\psi_{in}, 0\rangle)$
if $b = 1$
    FIND := 1
return $U_O(|\psi'_{in}, \psi_{out}\rangle)$

FindSolution$(\mathcal{T}, \tilde{x}, x)$
Parse $\mathcal{T}$ as $(y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)})$
Run SetTest$(y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)})$
    returning $\mathfrak{a}'$ s.t. $\mathfrak{a}' \diamond \tilde{x} = x_1$
return $\mathfrak{a}'$
```

Fig. 7. Reduction $\mathcal{B}$ for bounding the difference between G_0 and G_1 for the proof of Theorem 9. The function Solve solves for $\mathfrak{a}$ and outputs a set of (possibly multiple) solutions. The oracle $\mathrm{O}_\mathfrak{a}$ is simulated as in G_1. The simulation of $\mathrm{O}_{2\mathfrak{a}}$ is similar and explained in the proof.

which we can rearrange to

$$\underbrace{(1 + (-1)^{b_1} i - (-1)^{b_2} j)}_{=: d \in [-1,3]} \cdot \mathfrak{a} \equiv \mathfrak{d} - \mathfrak{c} \mod N \ .$$

If $\mathfrak{a}$ does not vanish, i.e., $d \neq 0$, we can extract one or multiple candidate solutions and check their correctness. If $d = 0$, we can simply simulate the oracle correctly by checking whether $\mathfrak{c} = \mathfrak{d}$. Also, note that in contrast to the above, it is not guaranteed that a solution for the congruence exists because we cannot assume that the input is a valid DDH tuple. In that case, the oracle outputs 0.

The simulation of the second decision oracle works similarly. Note that on input $|y, z\rangle$ the oracle is supposed to check whether $2\mathfrak{a} \diamond y = z$. Thus, if we get $|y_{(i,\mathfrak{c},b_1)}, z_{(j,\mathfrak{d},b_2)}\rangle$ and they indeed form a valid tuple, then it holds that

$$\underbrace{(2 + (-1)^{b_1} i - (-1)^{b_2} j)}_{=: d \in [0,4]} \cdot \mathfrak{a} \equiv \mathfrak{d} - \mathfrak{c} \mod N \ .$$

We proceed as above. If $d \neq 0$ and the input forms a valid tuple, we can extract the correct solution. If no solution exists, then the input cannot be a valid tuple and we output 0. If $d = 0$, we check whether $\mathfrak{c} = \mathfrak{d}$ and output the corresponding bit.

It remains to reduce G_1 to $\mathsf{GA\text{-}DLOG}$. That is,

$$\Pr[\mathrm{G}_1 \Rightarrow 1] \leq \epsilon_\mathcal{C} \ .$$

We construct a reduction $\mathcal{C}$ against $\mathsf{GA\text{-}DLOG}$ as follows. $\mathcal{C}$ gets as input the origin $\tilde{x}$ and a set element $\mathfrak{a} \diamond \tilde{x}$ and forwards both to the algebraic adversary $\mathcal{A}$. We simulate quantum queries to the decision oracles as described before.

At the end of the execution, $\mathcal{A}$ will output a solution (y_0, y_1, y_2) along with representations $(i_0, \mathfrak{s}_0, b_0)$, $(i_1, \mathfrak{s}_1, b_1)$ and $(i_2, \mathfrak{s}_2, b_2)$.

Assuming that $\mathcal{A}$ produces a valid solution, namely

$$y_1 = 2\mathfrak{a} \diamond y_0 \qquad \text{and} \qquad y_2 = -\mathfrak{a} \diamond y_0 \ ,$$

the following two equations hold:

$$\begin{aligned} \mathfrak{s}_1 + (-1)^{b_1} i_1 \mathfrak{a} &\equiv \ 2\mathfrak{a} + (-1)^{b_0} i_0 \mathfrak{a} + \mathfrak{s}_0 \mod N \\ \mathfrak{s}_2 + (-1)^{b_2} i_2 \mathfrak{a} &\equiv -\mathfrak{a} + (-1)^{b_0} i_0 \mathfrak{a} + \mathfrak{s}_0 \mod N \end{aligned}$$

Combining the two equations and solving for $\mathfrak{a}$ yields

$$\underbrace{(3 - (-1)^{b_1} i_1 + (-1)^{b_2} i_2)}_{=: d \in [1,5]} \cdot \mathfrak{a} \equiv \mathfrak{s}_1 - \mathfrak{s}_2 \mod N \ .$$

A solution to this congruence always exists since the above equation is a consequence of the (assumed) correctness of $\mathcal{A}$ and the prior choice of $\mathfrak{a}$. However, there could exist multiple solutions in case $d > 1$ and $\gcd(d, N)$ divides $\mathfrak{s}_1 - \mathfrak{s}_2$. Then, we can find the correct one by computing $\mathfrak{a}' \diamond \tilde{x}$ for all candidates $\mathfrak{a}'$ and compare it with the challenge. This concludes the proof of Theorem 9. $\square$