Quantum Complexity for Discrete Logarithms and Related Problems

Minki Hhan[*] KIAS

Takashi Yamakawa[†] NTT Social Informatics Laboratories

Aaram Yun[‡] Ewha Womans University

July 5, 2023

Abstract

This paper studies the quantum computational complexity of the discrete logarithm and related group-theoretic problems in the context of "generic algorithms"—that is, algorithms that do not exploit any properties of the group encoding.

We establish a generic model of quantum computation for group-theoretic problems, which we call the quantum generic group model, as a quantum analog of its classical counterpart. Shor's algorithm for the discrete logarithm problem and related algorithms can be described in this model. We show the quantum complexity lower bounds and (almost) matching algorithms of the discrete logarithm and related problems in this model. More precisely, we prove the following results for a cyclic group $\mathcal{G}$ of prime order.

- Any generic quantum discrete logarithm algorithm must make $\Omega(\log|\mathcal{G}|)$ depth of group operation queries. This shows that Shor's algorithm that makes $O(\log|\mathcal{G}|)$ group operations is asymptotically optimal among the generic quantum algorithms, even considering parallel algorithms.
- We observe that some (known) variations of Shor's algorithm can take advantage of classical computations to reduce the number and depth of quantum group operations. We introduce a model for generic hybrid quantum-classical algorithms that captures these variants, and show that these algorithms are almost optimal in this model. Any generic hybrid quantum-classical algorithm for the discrete logarithm problem with a total number of (classical or quantum) group operations Q must make $\Omega(\log|\mathcal{G}|/\log Q)$ quantum group operations of depth $\Omega(\log\log|\mathcal{G}| - \log\log Q)$. In particular, if $Q = \mathrm{poly}\log|\mathcal{G}|$, classical group operations can only save the number of quantum queries by a factor of $O(\log\log|\mathcal{G}|)$ and the quantum depth remains as $\Omega(\log\log|\mathcal{G}|)$.
- When the quantum memory can only store t group elements and use quantum random access memory (qRAM) of r group elements, any generic hybrid quantum-classical algorithm must make either $\Omega(\sqrt{|\mathcal{G}|})$ group operation queries in total or $\Omega(\log|\mathcal{G}|/\log(tr))$ quantum group operation queries. In particular, classical queries cannot reduce the number of quantum queries beyond $\Omega(\log|\mathcal{G}|/\log(tr))$.

As a side contribution, we show a multiple discrete logarithm problem admits a better algorithm than solving each instance one by one, refuting a strong form of the quantum annoying property suggested in the context of password-authenticated key exchange protocol.

[*]E-mail:minkihhan@kias.re.kr

[†]E-mail:takashi.yamakawa.obf@gmail.com

[‡]E-mail:aaramyun@ewha.ac.kr

Contents

1 Introduction

The discrete logarithm (DL) problem and related problems have long been fundamental cryptographic primitives in the pre-quantum world [DH76, Gam85]. However, the emergence of quantum computing has drastically altered the landscape of cryptography in the post-quantum world. Shor's algorithm [Sho94] has demonstrated that the DL problem (and integer factoring) can be solved in quantum polynomial time, rendering many cryptographic protocols that rely on the DL problem insecure against full-fledged quantum computers.

While algorithmic optimizations for quantum algorithms solving the DL problem have shown significant progress [ME98, GE21, PG14, Gid19, RNSL17, HJN$^+$20], the base algorithm of these circuit optimizations is essentially the same as Shor's original one or its near variants [Kal17, EH17, Eke21, Eke19]. The complexity of the quantum DL algorithms is still dominated by $O(\log|\mathcal{G}|)$ group operations for the underlying group $\mathcal{G}$, just like in the original Shor's algorithm. As such, no asymptotic improvements have been made since the original algorithm.

This state of affairs raises an important question:

Can we solve the discrete logarithm asymptotically better than Shor's algorithm?

There are several potential approaches to addressing this question. In an extreme case, a direct quantum algorithm with better asymptotic complexity may suddenly appear. Alternatively, a hybrid classical-quantum algorithm could take advantage of the potentially massive power of classical computation with a favorable classical-quantum tradeoff. Another interesting avenue of exploration is a shallow quantum circuit that exploits parallelism, making quantum depth another important measure of complexity. Cleve and Watrous [CW00] showed a lower bound on the depth for the quantum Fourier transform, which is a crucial step of Shor's algorithm. However, there might exist a completely different quantum algorithm that does not rely on the quantum Fourier transform.[1] To the best of our knowledge, there is no known lower bound, in terms of either time complexity or depth, for the quantum complexity of the DL problem.

1.1 This Work

In this paper, we study the hardness of the discrete logarithm problem and related problems by considering a natural class of quantum algorithms referred to as generic algorithms. A generic quantum algorithm is an algorithm that does not take advantage of the special properties of the encodings of group elements. Instead, these algorithms only use group operations only in a black-box manner, potentially in superposition.

We formally establish the quantum generic group model (QGGM) by restricting that access to group elements is provided through the group oracle. The QGGM resembles the classical generic group model (GGM) [Sho97, Mau05] proposed for arguing the security of group-theoretic cryptographic problems in classical settings. As in the classical GGM, the main complexity measure in the QGGM is the number of group operation queries. In addition, we are also concerned with the *depth* of group operation queries to study the power of near-term quantum computers.

Lower bound in fully quantum setting. Our first result states that no generic quantum algorithm in the QGGM can solve the DL problem much faster than Shor's original algorithm, even with parallel group operations. Precisely, we show the following theorem.

[1] Technically, the phase estimation-based DL algorithm [Kit96] can be done without the quantum Fourier transform.

Theorem 1.1. *For a prime-order cyclic group $\mathcal{G}$, any generic quantum algorithm solving the DL problem over $\mathcal{G}$ must make queries of depth $\Omega(\log|\mathcal{G}|)$.*

To establish this theorem, for any generic quantum DL algorithm A, we construct a generic *classical* DL algorithm B in the GGM that perfectly simulates the output of A. Although the classical simulation may require unbounded time for precise simulation, its query complexity is only exponentially larger than that of A. It is known that a generic DL algorithm in the classical GGM must make $\Omega(|\mathcal{G}|^{1/2})$ classical group operation queries even if the algorithm is allowed to run in unbounded time [Sho97, Mau05]. Combined with the above simulation with an exponential blowup, we obtain the desired result. We also establish the similar hardness of other group-theoretic problems, such as CDH and DDH using this simulation. We note that the naïve version of Shor's algorithm[2] has the matching group operation complexity to the lower bound in Theorem 1.1.

Hybrid quantum-classical algorithms. The above result may initially seem sufficient to refute our main question. However, this is not the case because this lower bound only considers purely quantum algorithms. We observe that some simple (combination of) folklore hybrid quantum-classical algorithms can do better than the purely quantum bound, exploiting classical computation to perform most group operations.

These hybrid algorithms consist of two phases: They first compute multiple group elements using $O(\text{polylog}\,|\mathcal{G}|)$ classical group operation queries and store them as precomputed data. Then, they implement Shor's algorithm using the stored group elements using $O(\log|\mathcal{G}|/\log\log|\mathcal{G}|)$ quantum group operations and $O(\log\log|\mathcal{G}|)$ quantum query depth (Theorems 7.2 and 7.3).

We complement these algorithms by proving the matching lower bounds. We formalize a model for generic hybrid quantum-classical algorithms that captures the above algorithms and more general class of algorithms. In the model, we allow an algorithm to make both classical and quantum group operation queries with the restriction that it is *forced to measure* all the registers whenever its quantum query number or depth count exceeds a certain threshold. It is supposed to capture hybrids of classical and quantum computers with limited coherence time. Note that we do not consider noises in our model whereas actual near-term quantum computers are likely to be noisy. Since our main results are the lower bounds, this just makes our results stronger.

The following theorem states the limitations of the generic hybrid algorithms, showing that the above hybrid algorithms are indeed optimal with respect to both query number and depth.

Theorem 1.2 (Informal)**.** *For a prime-order cyclic group $\mathcal{G}$, any generic hybrid quantum-classical algorithm solving the DL problem with $O(\text{poly}\log|\mathcal{G}|)$ total queries (including both classical and quantum) must make $\Omega(\log|\mathcal{G}|/\log\log|\mathcal{G}|)$ quantum queries of depth $\Omega(\log\log|\mathcal{G}|)$ between some two consecutive forced measurements.*

More generally, any generic hybrid DL algorithm with Q total queries must make $\Omega(\log|\mathcal{G}|/\log Q)$ quantum queries of depth $\Omega(\log\log|\mathcal{G}| - \log\log Q)$ between some two consecutive forced measurements.

Quantum memory-bounded algorithms. Quantumly processable memory is an expensive resource, either quantum memory that can store quantum states or quantum random accessible (classical) memory (qRAM) that stores classical data but can be accessed coherently.[3] While the

[2] We describe Shor's algorithm in the QGGM in Section 7 for completeness, with the other algorithms in the QGGM.
[3] Formally, it enables one to realize a unitary $|i\rangle \otimes |0\rangle \mapsto |i\rangle \otimes |x_i\rangle$ for a classical data $(x_i)_i$.

original Shor's algorithm only uses quantum memory that stores a single group element, the hybrid algorithms make use of relatively large quantum memory (Theorem 7.3) or large qRAM (Theorem 7.2). This motivates the question of whether quantumly processable memory is necessary even for a mild speed-up of Shor's algorithm.

We prove that it is indeed necessary. The following theorem asserts such a lower bound.

Theorem 1.3. *For a prime-order cyclic $\mathcal{G}$, any generic hybrid algorithm solving the DL problem with quantum memory that can store t group elements and no qRAM must make either $\Omega(\sqrt{|\mathcal{G}|})$ classical or quantum group operation queries in total or $\Omega(\log|\mathcal{G}|/\log t)$ quantum group operation queries between some two consecutive forced measurements.*[4]

More generally, any generic hybrid DL algorithm with quantum memory that can store t group elements and qRAM that can store r group elements must make either $\Omega(\sqrt{|\mathcal{G}|})$ group operation queries in total or $\Omega(\log|\mathcal{G}|/\log(tr))$ quantum group operation queries between some two consecutive forced measurements.

In particular, the above theorem means that classical queries cannot reduce the number of quantum queries beyond $\Omega(\log|\mathcal{G}|/\log t)$, or just $\Omega(\log|\mathcal{G}|)$ when $t = O(1)$. We have algorithms that match the above lower bounds: Baby-step giant-step algorithm makes $O(\sqrt{|\mathcal{G}|})$ classical group operations, and the hybrid algorithm in Theorem 7.2 with quantum memory that can store t group elements and no qRAM makes $\Omega(\log|\mathcal{G}|/\log t)$ quantum queries.

The multiple DL problem. The multiple discrete logarithm problem asks to solve multiple instances of the DL problem with the same underlying group simultaneously. When m DL instances are given, this problem is written by m-MDL. This problem is particularly interesting in the context of the standard curves in elliptic curve cryptography, where only a few curves are recommended as standard. In the classical setting, Kuhn and Struik [KS01] suggested an $O(\sqrt{m|\mathcal{G}|})$ generic algorithm for the m-MDL problem, and Yun [Yun15] proved the matching lower bound.

In Theorem 7.5, we present a generic quantum MDL algorithm using the results in vectorial addition chain [Pip80]. If $\log m/\log|\mathcal{G}| = o(1)$ and $m = \Omega(\log|\mathcal{G}|)$, it solves the m-MDL problem using $O(m\log|\mathcal{G}|/\log(m\log|\mathcal{G}|))$ group operations. This gives an amortized group operation complexity of $O(\log|\mathcal{G}|/\log m)$ per DL instance.

Regarding Theorem 1.1, the complexity of the m-MDL problem is lower than solving each instance individually. It is related to the quantum annoying property [Tho19, ES21] suggested in the context of password-authenticated key exchange (PAKE), which roughly means that quantum algorithms must solve a DLP for each password guess of PAKE. Our algorithm shows that the strongest form of quantum annoying cannot hold, regardless of the PAKE scheme construction.

We can derive the lower bound of the m-MDL problem similarly to Theorem 1.1 and using the classical lower bound given in [Yun15]. However, this would only give a lower bound of $\Omega(\log m + \log|\mathcal{G}|)$ group operations. So there is an apparent gap between the upper and lower bounds. We leave more accurate generic asymptotic complexity of m-MDL as an open problem.

1.2 Discussion

Tight complexity. Our lower bounds show asymptotically tight group operation complexity, but the constant factor has room for improvement. In the formal theorems, the concrete quantum query bounds are $0.25\log|\mathcal{G}| + O(1)$ (or depth) in the fully quantum case (Theorem 4.2) and

[4]This gives a depth lower bound of $\Omega(\log|\mathcal{G}|/t\log t)$ as an immediate corollary as an algorithm can make at most t queries in one parallel query in this setting.

$0.5 \log |\mathcal{G}| + O(1)$ for the memory bounded hybrid case with $t = r = 1$ (Theorem 6.2). Shor's DL algorithm and early variants [ME98, Kit96] make quantum and classical group operations $2 \log |\mathcal{G}|$ times each, having a gap in the constant factor.

The hybrid quantum-classical algorithms [Kal17, Eke19, Eke21, EH17] narrow down this gap. These algorithms solve the DL problem by repeating a certain procedure with $\log |\mathcal{G}| + O(1)$ group operations about $\log^{O(1)} |\mathcal{G}|$ times,[5] with appropriate classical pre- and post-processing. The constant gap still exists besides the number of subroutine calls. Filling this gap is of practical interest.

Another interesting tradeoff point in our lower bound is the hybrid case (without memory bound) in Theorem 5.2. We may ask if a small number of quantum group operations could reduce the classical group operation queries. This theorem says that if a generic hybrid algorithm makes a single quantum group operation, then it should make $\Omega(|\mathcal{G}|^{0.25})$ classical group operations. In other words, this does not rule out a hybrid DL algorithm with $|\mathcal{G}|^{0.25}$ classical group operations and a single quantum group operation, which we do not know how to do. Theorem 6.2 rules out this case if there is a memory constraint.

The quantum complexity of the composite-order DL problem is also unknown. We do not know how to use the composite order either in constructing algorithms or proving lower bounds.

Generic vs. non-generic algorithms. While many group-theoretic algorithms discussed above can be viewed as generic quantum algorithms, some variants leverage specific encoding structures [PZ03, HS05, RS14, RNSL17, HJN+20] for theoretic or practical purposes. In particular, Høyer and Spalek [HS05] showed that the DL problem on $\mathbb{Z}_N$ can be solved by a hybrid quantum-classical algorithm with a constant quantum depth if we allow for unbounded fan-out gates.[6] This overcomes our quantum depth lower bound in Theorem 1.2.[7] This is possible because their algorithm is non-generic. For example, they use that multiplication of many elements of $\mathbb{Z}_N$ can be done in $\mathbf{TC}_0$, i.e., computed by a constant depth classical circuit with threshold gates [SBKH93].

This circumstance is reminiscent of the classical GGM, where some non-generic algorithms, such as index calculus, show better efficiency than generic algorithms by exploiting the integer encoding of group elements. Still, the classical GGM has been used as a meaningful model for arguing the hardness of group-theoretic problems, especially for the general elliptic curves. Thus, we believe that lower bounds in the QGGM are at least as meaningful as those in classical GGM. Moreover, to the best of our knowledge, all non-generic quantum algorithms for the DL problem are circuit optimization of (variants of) Shor's algorithm, which is generic, while non-generic classical algorithms start from fundamentally different ideas from generic algorithms. This fact gives us more motivation to study the limitations of generic quantum algorithms.

Maurer-style vs. Shoup-style QGGM. In the classical setting, there are two formalizations of the GGM, one by Shoup [Sho97] and the other by Maurer [Mau05]. In Shoup's GGM, generic algorithms are given random labels of group elements and can perform group operations by sending labels to the oracle. On the other hand, in Maurer's GGM, all group elements are kept by the oracle, and generic algorithms can access to them only through group operation or equality check

[5]Precisely, Kaliski's algorithm [Kal17] repeats a subroutine of $\log |\mathcal{G}| + 1$ group operations $O(\log^{1.5} |\mathcal{G}|)$ times, and Ekerå's algorithm repeat subroutines with $(1 + 1/s) \log |\mathcal{G}|$ group operations about s times for some bounded s [Eke21].

[6]It does not contradict the depth lower bound of the quantum Fourier transform [CW00], which assumes that each gate acts on a constant number of qubits.

[7]Using fan-out gates does not affect the query depth in the QGGM.

queries. These two GGMs are known to be equivalent for "single-stage games," which include the DL and related problems considered in this paper [Zha22].[8]

Our QGGM is defined as a quantum analog of Maurer's GGM. It is possible to define it in Shoup's style. Indeed, such a model was already considered in [Zha21] under the name of "post-quantum GGM." It is easy to show that any generic algorithm that works in our (Maurer-style) QGGM also works in Shoup-style QGGM. On the other hand, it seems difficult to show the other direction in the quantum setting, even if we focus on single-stage games. Thus, it would make our results stronger if we could prove similar lower bounds in Shoup-style QGGM. We believe that the lower bound in the fully quantum setting (Theorem 1.1) can be extended to Shoup-style QGGM with a similar proof if the label space is much larger than the group order. On the other hand, we do not know how to generalize the lower bounds for hybrid algorithms (Theorems 1.2 and 1.3) to Shoup-style QGGM. For this reason, we focus on Maurer-style QGGM in this paper. We believe that lower bounds in Maurer-style QGGM are still meaningful, given that it captures Shor's algorithm and many variants.

Hidden subgroup problems and other potential directions. This paper suggests the number of (quantum) group operations as a complexity measure for studying the DL and related problems. We discuss the potential applications to the hidden subgroup problem (HSP).

In the hidden subgroup problem (HSP) literature, the primary complexity measure is the query complexity to the oracle function hiding a subgroup. The standard approach, or *Fourier sampling*, to the HSP over abelian groups makes a single oracle query. This approach is also a subroutine to solve some HSPs over non-abelian groups, e.g., in [HRT00, EH00, Kup05]. Finally, it is shown that $O(\log^4|\mathcal{G}|)$ queries suffice for the HSP over an arbitrary group [EHK04]. This makes proving lower bounds in terms of query complexity unlikely to yield meaningful results.[9]

Interestingly, these HSP algorithms can be considered generic algorithms by extending our QGGM for general groups. Also, contrary to the query complexity, the group operation complexity of [EHK04] is exponentially large. One may wonder if the group operation complexity can provide an interesting lower bound of the HSP for some nonabelian groups. The full answer is elusive with this paper's tools. The *dihedral group* case, a crucial case regarding its connection to the lattice-based [Reg04] and isogeny-based cryptography [Pei20, CJS14], has a negative answer to this question, as the algorithm of Ettinger and Høyer [EH00] only makes a polynomial number of group operations.

We believe exploring other potential applications of the generic model presented in this paper is an interesting topic. For example, can we argue something about factoring by extending our model to the ring operations?

1.3 Related Works

Post-quantum GGM. Zhandry [Zha21] introduced a model called post-quantum GGM as a quantum analog of Shoup's GGM. He showed that the generic group oracle in the model is quan-

[8]Jager and Schwenk [JS08] originally claimed a general equivalence, but Maurer, Portmann, and Zhu [MPZ20] pointed out a counterexample. Zhandry [Zha22] resolved this issue by reproving the equivalence in the case of single-stage games. Precisely speaking, he proved equivalence between Shoup's GGM and what is called the type-safe model, which is a variant of Maurer's GGM for single-stage games, but there is no difference between the type-safe model and Maurer's GGM when we consider group-theoretic problems such as the DL problem.

[9]For a certain restricted class of algorithms, there are some known limitations [MRS08, HMR$^+$10].

tumly reset indifferentiable from ideal ciphers. This means that generic groups can be used to construct symmetric key encryption secure against quantum adversaries. On the other hand, the work does not discuss the hardness of the DL and its related problems in the post-quantum GGM.

Generic group action model. While the DL problem on cyclic groups can be solved in quantum polynomial time by Shor's algorithm, the DL problem for *group actions* is believed to be hard against quantum computers. Such group actions with the quantum hardness of the DL problem have been used as bases of some proposals of post-quantum cryptography [Cou06, RS06, CLM+18, JQSY19]. Montgomery and Zhandry [MZ23] and Duman et al. [DHK+23] introduced generic models for group actions and studied the relations between the DL and related problems. We stress that their results are not proving the lower bounds.[10]

Hybrid quantum-classical algorithms. The hybrid quantum-classical algorithms have recently begun to attract more attention in various aspects. In [CCL23, CM20], the authors studied the relations between the hybrid algorithm with shallow quantum circuits and BQP, refuting the conjecture of Josza [Joz06] and proving Aaronson's conjecture [Aar05]. The study of hybrid algorithms with shallow quantum circuits was continued in [ACC+22] relative to random oracles. [Ros22] studied the hybrid algorithm in the context of Grover's algorithm, showing that classical queries cannot assist quantum computation. [HLS22] further developed the tools for hybrid algorithms with random oracles and showed a similar result for collision finding. Our model of generic hybrid algorithms is inspired by [CCL23, ACC+22], as well as the other papers.

2 Technical Overview

Classical GGM. First, we recall the classical GGM as formalized by Maurer [Mau05]. Let $\mathcal{G}$ be a cyclic group of order N with a generator g in which we consider group-theoretic problems such as the DL problem. A generic algorithm A is formalized as an oracle-aided algorithm that has classical access to an oracle, which keeps a table T storing elements of $\mathbb{Z}_N$. At the beginning, when A takes $g^{y_1}, ..., g^{y_m}$ as input, the table T is initialized as $(y_1, ..., y_m, 0, ..., 0)$.[11] The generic algorithm A can make the following two types of queries:

- *Group operation queries.* When A submits $(b, i, j, k) \in \{0, 1\} \times \mathbb{N}^3$, the oracle finds i-th element x_i and j-th element x_j in the table T and overwrites the k-th element of T by $x_i + (-1)^b x_j$. Nothing is returned to A.
- *Equality queries.* When A submits $(i, j) \in \mathbb{N}^2$, the oracle returns 1 if i-th and j-th elements of T are equal and otherwise returns 0.

We only count the number of group operation queries and allow equality queries for free, following the previous models [Mau05, Zha22].[12]

Finally, A outputs a bit string or an index i^* of T. In the latter case, $g^{x_{i^*}}$ is treated as A's output where x_{i^*} is the i^*-th element in T.

[10]Indeed, [DHK+23] argued that we could not hope for the superpolynomial lower bound of the DL problem in group actions due to [EH00], similar to our discussion on the dihedral HSP.
[11]The size of T can be unbounded.
[12]See Section 3 for more discussion.

Quantum GGM. We define the Quantum GGM (QGGM) as a natural quantum analog of the classical GGM where A is allowed to make quantum queries and the table T is stored in a quantum register $\mathbf{T}$. However, since overwriting values of quantum registers is not unitary, we formalize group operation queries in a slightly different way. Specifically, a group operation query is (a superposition of) $(b, i, j) \in \{0, 1\} \times \mathbb{N}^2$ and the oracle replaces i-th element of the table register $\mathbf{T}$ with $x_i + (-1)^b x_j$ (in superposition) where x_i and x_j are i-th and j-th elements of $\mathbf{T}$ before the query, respectively. In this way, we can ensure that it is a unitary operation. For clarity, we describe how the oracle works for group operation and equality queries where $\mathbf{Q}$ is the query register:

- *Group operation queries.* Apply the following unitary on $\mathbf{Q}$ and $\mathbf{T}$:

$$|b, i, j\rangle_{\mathbf{Q}} \otimes |..., x_i, ..., x_j, ...\rangle_{\mathbf{T}} \mapsto |b, i, j\rangle_{\mathbf{Q}} \otimes \left|..., x_i + (-1)^b x_j, ..., x_j, ...\right\rangle_{\mathbf{T}}$$

  if $i \neq j$ and otherwise it does nothing.

- *Equality queries.* Apply the following operation on $\mathbf{Q}$ and $\mathbf{T}$:

$$|b, i, j\rangle_{\mathbf{Q}} \otimes |..., x_i, ..., x_j, ...\rangle_{\mathbf{T}} \mapsto |b \oplus t, i, j\rangle_{\mathbf{Q}} |..., x_i, ..., x_j, ...\rangle_{\mathbf{T}}$$

  where $t = 1$ if $x_i = x_j$ and $t = 0$ otherwise.

Initialization and finalization of a generic algorithm are exactly the same as in the classical GGM except that $\mathbf{T}$ is measured in the computational basis at the end.

Basic idea: the fully quantum setting. Our idea is to simulate a generic algorithm A taking m group elements as input in the QGGM by a generic algorithm B in the *classical GGM* with an exponential blowup in the number of group operations (or simply, queries). Since we have a group operation complexity lower bound of $\Omega(|\mathcal{G}|^{1/2})$ for the DL problem in the classical GGM, such a simulation gives a lower bound of $\Omega(\log |\mathcal{G}|)$ in the QGGM. In particular, the classical lower bound holds even for unbounded algorithms as long as the condition on the query number is satisfied.

The idea for the simulation is extremely simple. At the beginning, the table register $\mathbf{T}$ of the QGGM has m non-zero elements $(y_1, ..., y_m)$. Suppose that A makes one (potentially parallel) quantum group operation query. After the query, $\mathbf{T}$ can only contain elements of the form $z_1 y_1 + z_2 y_2 + ... + z_m y_m$ where $|z_i| \leq 1$ for all $i \in [m]$ in any branch with a non-zero amplitude. After A makes the next (potentially parallel) quantum group operation query, a similar argument shows that $\mathbf{T}$ can only contain elements of the form $z_1 y_1 + z_2 y_2 + ... + z_m y_m$ where $|z_i| \leq 2$ for all $i \in [m]$. By repeating a similar argument recursively, one can see that after d-layer of parallel quantum group operation queries, $\mathbf{T}$ can only contain elements of the form $z_1 y_1 + z_2 y_2 + ... + z_m y_m$ where $|z_i| \leq 2^{d-1}$ for all $i \in [m]$. In particular, the number of such elements is at most $(2^d + 1)^m \leq 2^{m(d+1)}$. Thus, if the generic algorithm B in the classical GGM creates all these elements in its table in advance using $2^{m(d+1)}$ classical queries, it can perfectly simulate $\mathbf{T}$ for A. Note that B can run in unbounded time though it only makes only classical queries. In particular, it can simulate any quantum superposition of the group elements in the table by brute force. This means that a generic algorithm of query depth d in the QGGM can be perfectly simulated by a generic algorithm that makes $2^{m(d+1)}$ queries in the classical GGM. In particular, for the DL problem, we have $m = 2$ since the input is $y_1 = g$ and $y_2 = g^x$ for random x. Combined with the lower bound in the classical GGM, we obtain Theorem 1.1.

Hybrid quantum-classical algorithms. We explain how to extend the above idea to the hybrid quantum-classical algorithms. First, we describe our formalization of hybrid quantum-classical algorithms in the QGGM. A hybrid quantum-classical algorithm is characterized as a consecutive execution of quantum *subroutines* $U_1, ..., U_T$ followed by a classical post-processing algorithm A_{T+1}. Each subroutine U_i makes arbitrarily many classical queries and a bounded number or depth of quantum queries and measures all the registers including the table register $\mathbf{T}$ at the end. A_{T+1} makes arbitrarily many classical queries and no quantum query.

We first consider the depth-bounded case where each subroutine can have quantum query depth at most d. Let Q be the total number of the hybrid algorithm's queries including both classical and quantum ones. The idea is similar to the basic case: The simulation algorithm in the classical GGM creates all the group elements that may appear in the table register $\mathbf{T}$.

We first analyze each subroutine and then apply an inductive argument. Suppose that a subroutine U_i is described as a sequence $(C_{i,0}, O_{i,1}, ..., C_{i,d-1}, O_{i,d})$ where each $C_{i,j}$ only makes classical queries and each $O_{i,j}$ makes one parallel quantum query. Let Q_i be the total number of queries made by U_i and let $c_{i,j}$ be the number of classical queries made by $C_{i,j}$. Let m_i be the number of non-zero elements stored in the table register $\mathbf{T}$ when U_i starts. For $j = 0, 1, ..., d-1$, let $S_{i,j} \subseteq \mathbb{Z}_N$ be the set of elements that appear in the table register $\mathbf{T}$ in some branch with a non-zero amplitude right before the application of $O_{i,j+1}$ and let $S_d \subseteq \mathbb{Z}_N$ be the set right after the application of O_d (before the forced measurement). It is easy to see that we have $|S_{i,0}| \leq m_i + 1 + c_{i,0}$, $|S_{i,j}| \leq 2|S_{i,j-1}|^2 + c_{i,j}$, and $|S_{i,d}| \leq 2|S_{i,d-1}|^2$. Thus, we have

$$|S_{i,d}| \leq 2^{2^d} \left(m_i + 1 + c_{i,0} + c_{i,1} + ... + c_{i,d-1}\right)^{2^d} \leq 2^{2^d} \left(m_i + Q_i + 1\right)^{2^d}$$

Then, by a similar argument to the basic case, a generic algorithm in the classical GGM can simulate the subroutine U_i by making at most $2^{2^d} \left(m_i + Q_i + 1\right)^{2^d}$ group operation queries. Moreover, it is easy to see that we have $m_i \leq m + Q_1 + ... + Q_{i-1}$ where m is the number of elements given as input since each classical or quantum query can increase at most one new element in the table. Then, if we let c be the number of classical queries by A_{T+1}, the total number of classical queries needed to simulate the whole execution is at most

$$2^{2^d}(m + Q_1 + 1)^{2^d} + 2^{2^d}(m + Q_1 + Q_2 + 1)^{2^d} + ... + c \leq Q + T \cdot 2^{2^d}(m + Q + 1)^{2^d}$$

where we use $Q_1 + Q_2 + ... + Q_T + c \leq Q$. That is, any generic hybrid algorithm with the total number of queries Q and bounded quantum query depth d can be simulated by a generic algorithm in the classical GGM that makes at most $Q + T \cdot 2^{2^d}(m + Q + 1)^{2^d}$ classical queries. Combined with the classical GGM lower bound, we obtain the depth-bounded part of Theorem 1.2.

Next, we consider the query-bounded case where each subroutine can make at most q quantum queries, and the total number of the hybrid algorithm's queries is Q, including both classical and quantum ones. The idea is similar to the depth-bounded case, but we have to count the number of elements that appear in the table register $\mathbf{T}$ more carefully by making use of the fact that there are no parallel queries. We use similar notations to the depth-bounded case where the difference is that each $O_{i,j}$ makes only one non-parallel group operation query instead of a parallel one and the index j ranges in $[q]$ instead of $[d]$. First, we remark that we can simulate $C_{i,0}, ..., C_{i,q-1}$ by making at most Q_i classical queries in an obvious way. Thus, we ignore them in the following analysis and simply add Q_i to the number of classical queries needed to simulate the subroutine at the end. Recall that m_i denotes the number of non-zero elements stored in $\mathbf{T}$ at the beginning of the subroutine U_i. Before the subroutine U_i applies $O_{i,1}$, $\mathbf{T}$ has a classical state that has at most $m_i + Q_i + 1$ elements including 0. After applying $O_{i,1}$, $\mathbf{T}$ is a superposition of at most $2(m_i + Q_i + 1)^2$ different tables. We observe that in each possible table with a non-zero amplitude, at most one new element is added. Thus, we can simulate $O_{i,1}$ by making at most $2(m_i + Q_i + 1)^2$ classical queries. Next, for each fixed branch of $\mathbf{T}$, we can do the same analysis to see that we can simulate $O_{i,2}$ by making at most $2(m_i + Q_i + 1)^2$ classical queries. Since there are $2(m_i + Q_i + 1)^2$ branches, the total number of classical queries needed to simulate $O_{i,2}$ is at most $2(m_i + Q_i + 1)^2 \cdot 2(m_i + Q_i + 1)^2 = 4(m_i + Q_i + 1)^4$ and we have at most $4(m_i + Q_i + 1)^4$ branches with a non-zero amplitude. By repeating a similar argument recursively, we can simulate $O_{i,j}$ by making $2^j(m_i + Q_i + 1)^{2j}$ classical queries. Thus, the total number of classical queries needed to simulate the subroutine $U_i = (C_{i,0}, U_{i,1}, ..., C_{i,q-1}, U_{i,q})$ is at most

$$Q_i + 2(m_i + Q_i + 1)^2 + ... + 2^q(m_i + Q_i + 1)^{2q} \leq Q_i + 2^{q+1}(m_i + Q_i + 1)^{2q}.$$

Noting that we have $m_i \leq m + Q_1 + ... + Q_{i-1}$, the total number of classical queries to simulate the whole hybrid algorithm is at most

$$\left(Q_1 + 2^{q+1}(m + Q_1 + 1)^{2q}\right) + \left(Q_2 + 2^{q+1}(m + Q_1 + Q_2 + 1)^{2q}\right) + ... + c \leq Q + T \cdot 2^{q+1}(m + Q + 1)^{2q},$$

where we use $Q_1 + Q_2 + ... + Q_T + c \leq Q$. That is, any generic hybrid algorithm with the total number of queries Q and bounded quantum query number q can be simulated by a generic algorithm in the classical GGM that makes at most $Q + T \cdot 2^{q+1}(m + Q + 1)^{2q}$ classical queries. Combined with the lower bound in the classical GGM, we obtain the query-bounded part of Theorem 1.2.

Quantum-memory-bounded algorithms. To formalize a quantum-memory-bounded generic algorithm, we divide the table register $\mathbf{T}$ into the quantum part $\mathbf{T}_Q$ and classical part $\mathbf{T}_C$. We restrict $\mathbf{T}_Q$ to store at most t elements whereas $\mathbf{T}_C$ can store arbitrarily many elements. A generic hybrid quantum-classical algorithm without qRAM cannot send a group operation query that involves a superposition over indices in $\mathbf{T}_C$. In this setting, the number of new elements that may be computed by one quantum query is at most $2t(t-1)$. Thus, by a similar analysis to the query-bounded case for quantum-memory-bounded generic algorithms in the previous paragraph, we can see that the number of classical queries to simulate each subroutine U_i is at most

$$Q_i + 2t(t-1) + ... + 2^q(t(t-1))^q \leq Q_i + 2^{q+1}(t(t-1))^q,$$

and thus the total number of classical queries to simulate the whole hybrid algorithm is at most

$$\left(Q_1 + 2^{q+1}(t(t-1))^q\right) + \left(Q_2 + 2^{q+1}(t(t-1))^q\right) + ... + c \leq Q + T \cdot 2^{q+1}(t(t-1))^q.$$

Combined with the classical lower bounds, this implies the former part of Theorem 1.3.

For capturing qRAM that can store r elements, we allow a generic hybrid algorithm to make a query involving a superposition over indices in $\mathbf{T}_C$ as long as the number of indices in $\mathbf{T}_C$ involved in the superposition is at most r. In this setting, the number of new elements that may be computed by one quantum query is at most $2t \cdot (t - 1 + r)$. Thus, by a similar analysis where we replace $2t(t-1)$ with $2t \cdot (t - 1 + r)$, the total number of classical queries to simulate the whole hybrid algorithm is at most

$$\left(Q_1 + 2^{q+1}(t \cdot (t + r - 1))^q\right) + \left(Q_2 + 2^{q+1}(t \cdot (t + r - 1))^q\right) + ... + c \leq Q + T \cdot 2^{q+1}(t \cdot (t + r - 1))^q.$$

Combined with the classical lower bound, this implies the latter part of Theorem 1.3.

3 The Adversarial Model

This section defines the model of *generic* adversaries for the discrete logarithm and related problems. Section 3.1 defines the generic group model for the classical and quantum adversaries and Section 3.2 summarizes the cryptographic problems such as the discrete logarithm and their lower bounds in the classical generic group model.

3.1 The Generic Group Models

Classical generic group model. We first review the *classical* generic group model (GGM) as defined in [Mau05]. A generic algorithm A in the GGM interacts with an oracle that keeps a function $T : \mathbb{N} \to \mathbb{Z}_N$ for some positive integers N. We often regard T as a table consisting of group elements, and we often refer to $T(i)$ by the i-th element in the table T. At the beginning, T is initialized as $T(i) := y_i$ for $i \in [m]$ and $T(i) := 0$ for all $i > m$ where $(y_1, ..., y_m) \in \mathbb{Z}_N^m$ is the input of A. A is allowed to make the following queries:

- *Group operation queries.* When A submits $(b, i, j, k) \in \{0, 1\} \times \mathbb{N}^3$, the oracle overwrites $T(k) := T(i) + (-1)^b T(j)$. Nothing is returned to A.
- *Equality queries.* When A submits $(i, j) \in \mathbb{N}^2$, the oracle returns 1 if $T(i) = T(j)$, and 0 otherwise.

Finally, A outputs a classical string or a special symbol group along with an integer i. In the latter case, $T(i)$ is treated as A's output.

When we discuss query complexity of A, we only count the number of group operation queries and allow it to make equality queries for free following [Mau05, Zha22]. Assuming the zero-cost equality query makes our result stronger, and in fact describes the practice more appropriately. We refer to a more detailed discussion in [Zha22, Remark 3.1].

Quantum generic group model. We extend the GGM to define the *quantum* generic group model (QGGM). A generic algorithm A in the QGGM works over a working register $\mathbf{W}$, a query register $\mathbf{Q}$, and a table register $\mathbf{T}$. The registers $\mathbf{W}$ and $\mathbf{Q}$ are initialized to be $|0...0\rangle$. The register $\mathbf{T}$ stores s group elements of $\mathbb{Z}_N$ for some positive integers s, N. At the beginning, $\mathbf{T}$ is initialized as $|y_1, ..., y_m, 0, ..., 0\rangle_{\mathbf{T}}$ where $(y_1, ..., y_m) \in \mathbb{Z}_N^m$ is the input of A. A can apply arbitrary quantum operations on $\mathbf{W}$ and $\mathbf{Q}$, but it can only act on $\mathbf{T}$ through the following types of queries:

- *Group operation queries.* Apply the following unitary $O_{\mathbf{Q},\mathbf{T}}$ on $\mathbf{Q}$ and $\mathbf{T}$:
$$|b, i, j\rangle_{\mathbf{Q}} \otimes |..., x_i, ..., x_j, ...\rangle_{\mathbf{T}} \mapsto |b, i, j\rangle_{\mathbf{Q}} \otimes \left|..., x_i + (-1)^b x_j, ..., x_j, ...\right\rangle_{\mathbf{T}} \tag{1}$$
if $i \neq j$ and otherwise it does nothing.
- *Equality queries.* Apply the following operation on $\mathbf{Q}$ and $\mathbf{T}$:
$$|b, i, j\rangle_{\mathbf{Q}} \otimes |..., x_i, ..., x_j, ...\rangle_{\mathbf{T}} \mapsto |b \oplus t, i, j\rangle_{\mathbf{Q}} |..., x_i, ..., x_j, ...\rangle_{\mathbf{T}}$$
where $t = 1$ if $x_i = x_j$ and $t = 0$ otherwise.

Finally, A outputs a classical string or a special symbol group along with an integer $i \in [s]$. In the latter case, $\mathbf{T}$ is measured and the i-th element in the measurement outcome is treated as the output of A.

As in the classical GGM, the *query complexity* of A is defined by the number of group operation queries, and the equality queries are considered as free.

Remark 1. The QGGM has several differences from the classical GGM besides allowing quantum queries. First, the group operation query takes two indices (i, j) instead of three indices (i, j, k). This modification is made because we cannot "overwrite" the k-th element in the quantum setting since that is a non-unitary operation. Second, we put an upper bound s for the size of the table. This is to capture the size of quantum memory available for the adversary.

We remark that in many theorems (Theorems 4.1 and 5.1), the memory size does not appear in the bounds, which means that these theorems give lower bounds for arbitrarily large quantum memory size. On the other hand, in Theorem 6.1 where we consider hybrid quantum-classical generic algorithms with memory restrictions, the bound depends on some relevant parameters.

Remark 2. As we will see later in Section 7, Shor's algorithm can be written as a generic algorithm in the QGGM. Interestingly, Shor's algorithm does *not* make use of the equality queries.

Parallel-query generic algorithms. We define *parallel-query* generic algorithms in the QGGM. A parallel-query generic algorithm A in the QGGM works similarly to that in the QGGM except that it has K query registers $\mathbf{Q}_1, ..., \mathbf{Q}_K$ for some positive integer K (referred to as the *query width*) and is allowed to make parallel queries as follows:

- *Parallel group operation queries.* Let $O_{\mathbf{Q}_k,\mathbf{T}}$ be a unitary that works as in Equation (1) where $\mathbf{Q}_k$ plays the role of $\mathbf{Q}$. Then apply the following operation on $\mathbf{Q}_1, ..., \mathbf{Q}_K$ and $\mathbf{T}$:

$$\bigotimes_{k\in[K]} |b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |x_1, ..., x_s\rangle_{\mathbf{T}} \mapsto \prod_{k\in[K]} O_{\mathbf{Q}_k,\mathbf{T}} \bigotimes_{k\in[K]} |b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |x_1, ..., x_s\rangle_{\mathbf{T}}$$

  if $i_k \notin \{i_{k'}\}_{k'\in[K]\setminus\{k\}} \cup \{j_{k'}\}_{k'\in[K]}$ for all $k \in [K]$ and otherwise it does nothing. (Intuitively, the above condition means that multiple queries should not write to the same register and if one of the queries writes to some register, then that register should not be used as a control register for another query. Note that this does not prohibit parallel queries that share the same control register.)

- *Parallel equality queries.* Apply the following operation on $\mathbf{Q}$ and $\mathbf{T}$:

$$\bigotimes_{k\in[K]} |b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |x_1, ..., x_s\rangle_{\mathbf{T}} \mapsto \bigotimes_{k\in[K]} |b_k \oplus t_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |x_1, ..., x_s\rangle_{\mathbf{T}}$$

  where $t_k = 1$ if $x_{i_k} = x_{j_k}$ and $t_k = 0$ otherwise.

We call the number of parallel group operation queries by the *query depth* of A.

Remark 3. We do not consider parallel queries that mix group operation and equality queries because such queries can be split into a parallel group operation query and a parallel equality query.

Remark 4. Strictly speaking, here we are extending the QGGM to deal with parallel queries; when K is fixed to 1, this model becomes the QGGM above. On the other hand, if we always measure the query register whenever the algorithm makes a query, it is not hard to see that the QGGM is equivalent to the classical GGM when we allow s to be arbitrarily large.

Convention. We often want to analyze problems defined for a *multiplicative* cyclic group $\mathcal{G}$ in the (Q)GGM with $N = |\mathcal{G}|$. In this case, we occasionally identify $x \in \mathbb{Z}_N$ and $g^x \in \mathcal{G}$ where g is a generator of $\mathcal{G}$. In particular, the generator $1 \in \mathbb{Z}_N$ is identified with $g \in \mathcal{G}$ and the zero element $0 \in \mathbb{Z}_N$ is identified with $1 \in \mathcal{G}$. We also often abuse notation to write $\mathcal{G}$ to mean the generic group oracle in the (Q)GGM. For example, a (parallel-query) generic algorithm for the DL problem is written as $A^{\mathcal{G}}(g, g^x)$.

3.2 Group-theoretic Problems

3.2.1 Problems

The Discrete Logarithm (DL) Problem. In the discrete logarithm problem, an element $x \leftarrow \mathbb{Z}_N$ is uniformly chosen at random. The first and second elements of the table T are initialized by $T(1) = 1$ and $T(2) = x$ so that the input to the algorithm is (g, g^x). The adversary is asked to output x. The advantage of the DL adversary $A^{\mathcal{G}}$ is defined as follows:

$$\mathsf{Adv}_{\mathsf{DL}}(A^{\mathcal{G}}) = \Pr_{x}\left[A^{\mathcal{G}}(g, g^x) \to x\right],$$

where the input g, g^x denotes the elements stored in the table T.

The Computational/Decisional Diffie-Hellman Problem. In the computational Diffie-Hellman problem (CDH), two elements $x, y \leftarrow \mathbb{Z}_N$ are randomly chosen. An instance (g, g^x, g^y) is given to the adversary as elements in T. The adversary is asked to compute g^{xy} in the table. The advantage of the CDH adversary $A^{\mathcal{G}}$ is defined as follows:

$$\mathsf{Adv}_{\mathsf{CDH}}(A^{\mathcal{G}}) = \Pr_{x,y}\left[A^{\mathcal{G}}(g, g^x, g^y) \to g^{xy}\right].$$

In the decisional Diffie-Hellman problem (DDH), three elements $x, y, r \leftarrow \mathbb{Z}_N$ are randomly chosen, and either (g, g^x, g^y, g^r) or (g, g^x, g^y, g^{xy}) is given to the adversary, as elements in T, and the adversary is asked to decide which is the case by outputting a decision bit $b \in \{0, 1\}$.

The advantage of the DDH adversary $A^{\mathcal{G}}$ is defined as follows:

$$\mathsf{Adv}_{\mathsf{DDH}}(A^{\mathcal{G}}) = \left|\Pr_{x,y,r}\left[A^{\mathcal{G}}(g, g^x, g^y, g^{xy}) \to 1\right] - \Pr_{x,y,r}\left[A^{\mathcal{G}}(g, g^x, g^y, g^r) \to 1\right]\right|.$$

The Multiple Discrete Logarithm Problem. In the m-multiple discrete logarithm problem (m-MDL), m elements $x_1, ..., x_m \leftarrow \mathbb{Z}_N$ are independently and uniformly chosen at random. The m-MDL problem instance $g^{x_1}, ..., g^{x_m}$ is given to the adversary, stored in the $2, ..., (m+1)$-th elements of T along with the first element g of T. The adversary is asked to find all of $x_1, ..., x_m$. The advantage of the m-MDL adversary $A^{\mathcal{G}}$ is defined as follows:

$$\mathsf{Adv}_{m-\mathsf{MDL}}(A^{\mathcal{G}}) = \Pr_{x_1,...,x_m}\left[A^{\mathcal{G}}(g, g^{x_1}, ..., g^{x_m}) \to (x_1, ..., x_m)\right].$$

3.2.2 Classical lower bounds

The following theorems state the lower bounds of the above problems in GGM.

Theorem 3.1 (GGM lower bound of DL/CDH/DDH [Mau05, Sho97]). *Let Q be a positive integer, and $\mathcal{G}$ be a prime-order cyclic group. For a generic algorithm $A^{\mathcal{G}}$ in GGM with Q queries and for any $* \in \{\mathsf{DL}, \mathsf{CDH}, \mathsf{DDH}\}$, it holds that*

$$\mathsf{Adv}_*(A^{\mathcal{G}}) = O\left(\frac{Q^2}{|\mathcal{G}|}\right).$$

In particular, any constant-advantage algorithm in the GGM solving the DL/CDH/DDH problem makes at least $\Omega(\sqrt{|\mathcal{G}|})$ queries.

Theorem 3.2 (GGM lower bound of MDL [Yun15]). *Let Q be a positive integer, and $\mathcal{G}$ be a prime-order cyclic group. For a generic algorithm $A^{\mathcal{G}}$ in GGM with Q queries for the m-MDL problem $\mathcal{G}$, it holds that*

$$\mathsf{Adv}_{m\text{-}\mathsf{MDL}}(A^{\mathcal{G}}) = O\left(\left(\frac{e(Q+m+1)^2}{2m|\mathcal{G}|}\right)^m\right).$$

In particular, any constant-advantage algorithm in the GGM solving the m-MDL problem makes at least $\Omega(\sqrt{m|\mathcal{G}|})$ queries.

Remark 5. We stress that the query complexity is the only complexity measure when showing the lower bounds. In particular, the lower bounds in Theorems 3.1 and 3.2 do apply for adversaries even with *quantum* or *unbounded* computational powers, as long as they only make T classical queries. This observation is essential for our result.

4 Quantum Lower Bounds in the QGGM

In this section, we prove the quantum lower bounds of the DL and related problems in the QGGM. Our main technical tool is the following simulation theorem.

Theorem 4.1. *Let $\mathcal{G}$ be a group. Suppose that a parallel-query generic algorithm $A^{\mathcal{G}}$ in the QGGM is given m group elements as input and has the query depth at most d. Then there exists a generic algorithm B in the classical GGM for $\mathcal{G}$, given the same inputs, which makes at most $2^{(d+1)m}$ queries to the oracle such that the output distribution of $B^{\mathcal{G}}(y)$ and $A^{\mathcal{G}}(y)$ are identical for any input y.*

The generic algorithm B may perform quantum or unbounded computation, but the group operations are all done classically. As observed in Remark 5, the GGM lower bounds apply to the algorithm B. With this observation in mind, the quantum lower bound of the DL problem in the QGGM is an immediate corollary of Theorem 4.1.

Theorem 4.2. *Let $\mathcal{G}$ be a prime-order cyclic group. Any constant-advantage generic quantum algorithm solving the DL problem makes queries of depth at least $\Omega(\log|\mathcal{G}|)$.*

More precisely, the following holds. Let d be a positive integer. For a QGGM algorithm $A^{\mathcal{G}}$ making quantum queries of depth d for the DL problem over $\mathcal{G}$, it holds that

$$\mathsf{Adv}_{\mathsf{DL}}(A^{\mathcal{G}}) = O\left(\frac{2^{4d}}{|\mathcal{G}|}\right).$$

Proof. A DL instance consists of two group elements (g, g^x) , thus $m = 2$ for the DL problem. For $m = 2$, any generic quantum algorithm $A^{\mathcal{G}}$ with a d query depth can be simulated by a generic classical algorithm $B^{\mathcal{G}}$ with $Q = 2^{2(d+1)}$ queries by Theorem 4.1. The advantage of $B^{\mathcal{G}}$ is bounded by $O(Q^2/|\mathcal{G}|)$ due to Theorem 3.1, which shows the desired result. $\square$

The quantum CDH/DDH lower bound can be proven similarly.

Theorem 4.3. *Let $\mathcal{G}$ be a prime-order cyclic group. Any constant-advantage generic quantum algorithm solving the CDH/DDH problem makes queries of depth at least $\Omega(\log |\mathcal{G}|)$.*

More precisely, the following holds. Let d be a positive integer. For a generic quantum algorithm $A^{\mathcal{G}}$ making quantum queries of depth d and for any $* \in \{\mathsf{CDH}, \mathsf{DDH}\}$, it holds that

$$\mathsf{Adv}_*(A^{\mathcal{G}}) = O\left(\frac{2^{8d}}{|\mathcal{G}|}\right).$$

4.1 Proof of Theorem 4.1

We return to the proof of the main theorem. The idea of the proof is to *simulate* the generic algorithm in the QGGM by using classical group operations in the GGM. The simulation algorithm exhaustively computes all branches of the original algorithm. This may take an unbounded time, but the query complexity is bounded, and only exponentially larger than that of A, which suffices for our purpose; again, the classical GGM lower bounds only consider the query complexity.

Proof of Theorem 4.1. Let A be a parallel-query generic algorithm in the QGGM with query depth d and query width K that takes m group elements $y_1, ..., y_m$ as input. Without loss of generality, we assume that $y_1, ..., y_m$ are not 0. Let $N = |\mathcal{G}|$. Then we construct a generic algorithm B in the GGM that simulates A as follows.

Initialization. For the simulation, the algorithm B makes use of a "labeling function"

$$L : \mathbb{Z}_N[Y_1, ..., Y_m] \to [N] \cup \{\perp\},$$

which is gradually updated during the simulation.[13] Here, $\mathbb{Z}_N[Y_1, ..., Y_m]$ denotes the m-variate polynomial ring over $\mathbb{Z}_N$ with indeterminates $Y_1, ..., Y_m$. Intuitively, when we have $L(f) = \ell \neq \perp$, it should be understood that we give a "label" $\ell \in [N]$ to $f(y_1, ..., y_m) \in \mathbb{Z}_N$. For this to be well-defined, we always make sure that $L(f) = L(g)$ if and only if $f(y_1, ..., y_m) = g(y_1, ..., y_m)$ for any f, g on which L is defined (i.e., takes a non-$\perp$ value).

The algorithm B initializes L as follows:

1. Set $L(0) \leftarrow [N]$.
2. For $i = 1, ..., m$, uniformly set $L(Y_i) \in [N]$ under the constraint that $L(Y_i) \neq L(0)$ and $L(Y_i) = L(Y_j)$ if and only if $y_i = y_j$. Note that B can do this because it can check if $y_i = y_j$ by making an equality query to the classical group oracle.
3. Set $L(f) := \perp$ for all $f \notin \{0, Y_1, ..., Y_m\}$.[14]

[13] L will take non-$\perp$ values only on polynomials of degree at most 1 throughout the simulation.

[14] There are infinitely many elements in $\mathbb{Z}_N[Y_1, ..., Y_m]$, but B does not explicitly record that L is defined to be $\perp$ on those inputs. The value of L is understood to be $\perp$ unless it is explicitly defined to be a non-$\perp$ value.

B also defines a set $S \subseteq \mathbb{Z}_N[Y_1, ..., Y_m]$ of polynomials on which the value of L is defined (i.e., not $\perp$). That is, $S := \{0\} \cup \{Y_i\}_{i \in \{1,...,m\}}$. The set S will be updated along with L to ensure that it is always the set consisting of polynomials on which the value of L is defined.

B creates the following state as a simulation of the initial state for A:

$$|0...0\rangle_{\mathbf{W},\mathbf{Q}_1,...,\mathbf{Q}_K} \otimes |L(Y_1), ..., L(Y_m), L(0), ..., L(0)\rangle_{\mathbf{T}}.$$

During the simulation, we keep the invariance that for any $\ell \in \mathbb{Z}_n$ that appears in the register T of any branch with non-zero amplitude, there is $f \in S$ such that $\ell = L(f)$. This is satisfied at this point since $S = \{0\} \cup \{Y_i\}_{i \in \{1,...,m\}}$.

Local operation. When A applies some operation on its local registers $\mathbf{W}, \mathbf{Q}_1, ..., \mathbf{Q}_K$, B also applies the same operation.

Parallel group operation query. Suppose A makes a parallel group operation query. Let $S_{\mathsf{pre}} := S$. (We introduce S_{pre} to record the set S at the point of making the query since we will update the set S during the simulation below.) Then B does the following. Informally, the first step updates L and S to include the group elements potentially appearing after the query, and the second step defines the group operations over the labels. B simulates the parallel group operation of A in the last step.

1. For each pair $(f, g) \in S_{\mathsf{pre}}^2$ (in arbitrary order), do the following:
   (a) Check if there is any $h \in S$ such that

   $$h(y_1, ..., y_m) = f(y_1, ..., y_m) + g(y_1, ..., y_m).$$

   Note that B can check this by making one group operation query for the RHS and many equality queries to the classical group oracle because $f(y_1, ..., y_m)$, $g(y_1, ..., y_m)$, and $h(y_1, ..., y_m)$ have been already generated in the table of the classical group oracle.

   - If there exists such h, it sets
   $$L(f + g) := L(h).$$
   Note that this is well-defined since the RHS does not depend on the choice of h.[15]
   - Otherwise, it uniformly sets $L(f + g) \leftarrow [N]$ under the constraint that $L(f + g) \neq L(h)$ for all $h \in S$.

   Then update $S \leftarrow S \cup \{f + g\}$.

   (b) Similarly define $L(f - g)$ and update $S \leftarrow S \cup \{f - g\}$.
2. For labels $(\ell, \ell') \in [N]^2$, define $\ell \pm \ell'$ as follows:
   - Check if there is $(f, g) \in S_{\mathsf{pre}}^2$ such that $\ell = L(f)$ and $\ell' = L(g)$.

[15] As already mentioned, we always ensure that $L(h) = L(h')$ if and only if $h(y_1, ..., y_m) = h'(y_1, ..., y_m)$ for all $h, h' \in S$.

– If they exist, define

$$\ell \pm \ell' := L(f \pm g).$$

Note that this is well-defined since the RHS does not depend on the choice of (f, g).[16]

– Otherwise, define $\ell \pm \ell' := \bot$.

3. Then B simulates the group operation oracle by using the above defined operations for labels. That is, it does the following: For each $k \in [K]$, let $\widetilde{O}_{\mathbf{Q}_k,\mathbf{T}}$ be the unitary that works as follows:

$$|b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |..., \ell_{i_k}, ..., \ell_{j_k}, ...\rangle_{\mathbf{T}} \mapsto |b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes \left|..., \ell_{i_k} + (-1)^{b_k}\ell_{j_k}, ..., \ell_{j_k}, ...\right\rangle_{\mathbf{T}}$$

if $i_k \neq j_k$ and otherwise it does nothing. Then apply the following operation on $\mathbf{Q}_1, ..., \mathbf{Q}_K$ and $\mathbf{T}$:

$$\bigotimes_{k\in[K]} |b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |\ell_1, ..., \ell_s\rangle_{\mathbf{T}} \mapsto \prod_{k\in[K]} \widetilde{O}_{\mathbf{Q}_k,\mathbf{T}} \bigotimes_{k\in[K]} |b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |\ell_1, ..., \ell_s\rangle_{\mathbf{T}}$$

if $i_k \notin \{i_{k'}\}_{k'\in[K]\setminus\{k\}} \cup \{j_{k'}\}_{k'\in[K]}$ for all $k \in [K]$ and otherwise it does nothing.

Parallel equality query. When A makes a parallel equality query, B applies the following unitary:

$$\bigotimes_{k\in[K]} |b_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |\ell_1, ..., \ell_s\rangle_{\mathbf{T}} \mapsto \bigotimes_{k\in[K]} |b_k \oplus t_k, i_k, j_k\rangle_{\mathbf{Q}_k} \otimes |\ell_1, ..., \ell_s\rangle_{\mathbf{T}}$$

where $t_k = 1$ if $\ell_{i_k} = \ell_{j_k}$ and $t_k = 0$ otherwise.

Finalization. If A outputs a classical string, B outputs the same string. If A outputs the special symbol group and an integer i, B measures $\mathbf{T}$. Let $\ell \in [N]$ be the i-th element in the measurement outcome. B finds $f \in S$ such that $L(f) = \ell$. Then B finds the index i' such that the i'-th element stores $f(y_1, ..., y_m)$ in the table kept by its own classical group oracle. (Such f and i' must exist by the definition of the simulation.) Then B outputs group and i'.

The above completes the description of B. It is easy to see that B perfectly simulates A. To see this, we consider a hybrid simulator B' that has an additional capability to directly see $y_1, ..., y_m$ and works as follows: B' simulates the group operation oracle for A as in the QGGM except that it first randomly chooses a random bijection $I : \mathbb{Z}_N \to [N]$ and records $I(z)$ instead of z in T for any group element z. When A makes a group operation query, B' applies I^{-1} to the relevant entries, applies the group operation, and then applies I again. It is easy to see that B' perfectly simulates A because the random bijection I just induces a basis change in T. Moreover, the ways of simulation by B and B' are perfectly indistinguishable from the view of A because we can regard B as doing the same as B' except that it samples the bijection I via lazy sampling through L.

We count the number of group operation queries made by B. To do so, we observe that B only generates group elements that can be generated by depth-d applications of the group operation.

[16]Suppose that $L(f) = L(f')$ and $L(g) = L(g')$. Then on input $(y_1, ..., y_m)$, f, f' and g, g' have the same image, respectively, which implies that $f \pm g$ and $f' \pm g$ also have the same image under that input. Thus, $L(f \pm g) = L(f' \pm g')$.

In particular, the group elements generated after the first query are of the form $z_1y_1 + ... + z_my_m$ for $|z_i| \leq 1$, because quantum group operations for $i = j$ is ignored by definition. Inductively, we can show that it only needs to generate group elements z of the form

$$z = z_1y_1 + ... + z_my_m$$

where $|z_j| \leq 2^{d-1}$ for all $j = 1, ..., m$. These group elements can be generated in the table of the classical group oracle by making $(2^d + 1)^m \leq 2^{m(d+1)}$ group operation queries.

Combining the above arguments, we conclude that the classical GGM algorithm B can perfectly simulate the algorithm A by making at most $2^{(d+1)m}$ classical group operation queries. □

5 Hybrid Quantum-Classical Algorithms

This section presents the lower bounds of generic hybrid quantum-classical algorithms for group-theoretic problems. In Section 5.1, we formalize the model of generic hybrid algorithms. The lower bounds are presented in Section 5.2 using the hybrid simulation theorem, which is proved in Section 5.3.

5.1 The Model of Hybrid Algorithms

We establish the model of generic hybrid algorithms in this section. First, we define a *classical group operation* $O^C_{\mathbf{Q},\mathbf{T}}$ in the QGGM, which is illustrated in Figure 1, as follows.

1. Measure the query register $\mathbf{Q}$.
2. If the measurement outcome is b, i, j, measure the i-th and j-th entries of the register $\mathbf{T}$.
3. Apply $O_{\mathbf{Q},\mathbf{T}}$.

Intuitively, when we apply the classical group operations, all the relevant registers contain classical information. A non-classical group operation query is called *quantum*.

Remark 6. The classical group operation in the QGGM differs from the group operation in the GGM because the group oracles in the two models have different interfaces. In this section, an algorithm A (and its components to be described below) is always a generic algorithm in the QGGM, making classical or quantum group operation queries in the QGGM. On the other hand, an algorithm B is always an algorithm in the GGM, making group operation queries in the GGM.

In our generic hybrid model of computation, a hybrid algorithm is defined by a generic algorithm in the QGGM that by itself performs only classical group operations, but has access to a *quantum subroutine*, which is a generic quantum algorithm with a bounded number of quantum group operations but making an arbitrary number of classical group operations.

A quantum subroutine formalizes a quantum algorithm with a limited coherence time. More precisely, we define a q-query quantum subroutine by a generic quantum algorithm in the QGGM with at most q *quantum* group operations. After the q-th quantum group operation, it is forced to measure all registers on a computational basis. We call this measurement as the *forced measurement*. On the other hand, the quantum subroutine can make an arbitrary number of *classical* group operations. The total number of queries is the summation of classical and quantum queries. The

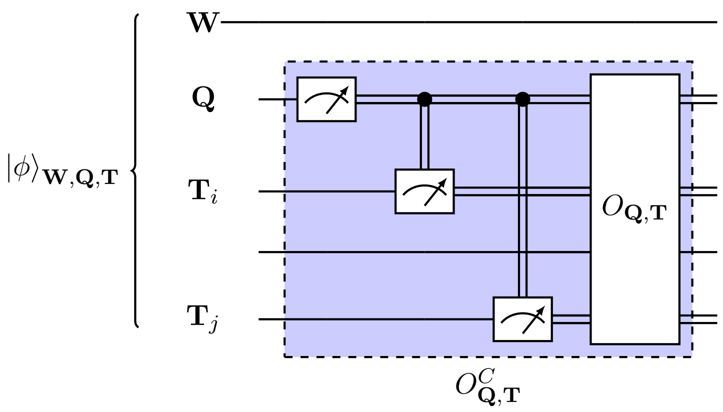
$|\phi\rangle_{\mathbf{W},\mathbf{Q},\mathbf{T}}$
$\mathbf{W}$
$\mathbf{Q}$
$\mathbf{T}_i$
$\mathbf{T}_j$
$O_{\mathbf{Q},\mathbf{T}}$
$O^C_{\mathbf{Q},\mathbf{T}}$

Figure 1: The classical group operation $O^C_{\mathbf{Q},\mathbf{T}}$: The single-line wires stand for quantum wires, while the double-line wires are for classical wires. $\mathbf{T}_i, \mathbf{T}_j$ denote the i-th and j-th entries of $\mathbf{T}$. We assume that the measurement outcome of $\mathbf{Q}$ indicates the i-th and j-th entries in this diagram. Recall that $O_{\mathbf{Q},\mathbf{T}}$ is a group operation query.

quantum subroutine can perform an arbitrary quantum map over the registers $(\mathbf{W}, \mathbf{Q})$ in the middle of its execution. We illustrate the rough behavior of the table register in a quantum subroutine as in Figure 2. As in the diagram, a quantum subroutine can be described by an alternating sequence of generic algorithms $(C_0, O_1, ..., C_{q-1}, O_q)$, where C_j is a generic algorithm in the QGGM that may include multiple classical group operations and O_j is a generic algorithm in the QGGM that includes a single quantum group operation for each j. Let c_j be the number of classical group operations C_j made.

Finally, a generic hybrid algorithm A is specified by a tuple $(U_1, ..., U_T, A_{T+1})$ of T quantum subroutines and a follow-up generic algorithm that is connected as in Figure 3. Here, A_{T+1} is a generic algorithm that only makes classical group operations, and U_j is a q-query quantum subroutine for each j. Again, the generic hybrid algorithm can perform an arbitrary quantum map over $(\mathbf{Q}, \mathbf{W})$ in the middle. Recall that the quantum subroutine makes the forced measurements for all registers including T on a computational basis, and the outcome of forced measurements will be given to the next subroutine (or A_{T+1}) as input.

Parallel-query generic hybrid algorithms. We also define *parallel-query* generic hybrid algorithm in the QGGM, by allowing quantum subroutines to be parallel-query generic algorithms in the QGGM. While a parallel classical group operation can be defined naturally, we only consider a classical group operation as defined above, which makes the simulation easier. Concretely, we separately use the parallel-query group operation oracle and the classical group operation oracle. The total number of queries is the summation of classical group operations and the query width K times the number of parallel-query quantum group operations.

A d-depth parallel quantum subroutine in the QGGM by a parallel-query generic quantum algorithm in the QGGM such that the number of parallel quantum group operation queries is bounded by d. Again, the quantum subroutines can make intermediate classical queries, and can apply any quantum map on $(\mathbf{Q}, \mathbf{W})$. Similarly with Figure 2, the d-depth parallel quantum subroutine can be described by a sequence of algorithms $(C_0, O_1, ..., C_{d-1}, O_d)$, where C_j is a generic algorithm with c_j classical group operations and O_j is a generic quantum algorithm with a single

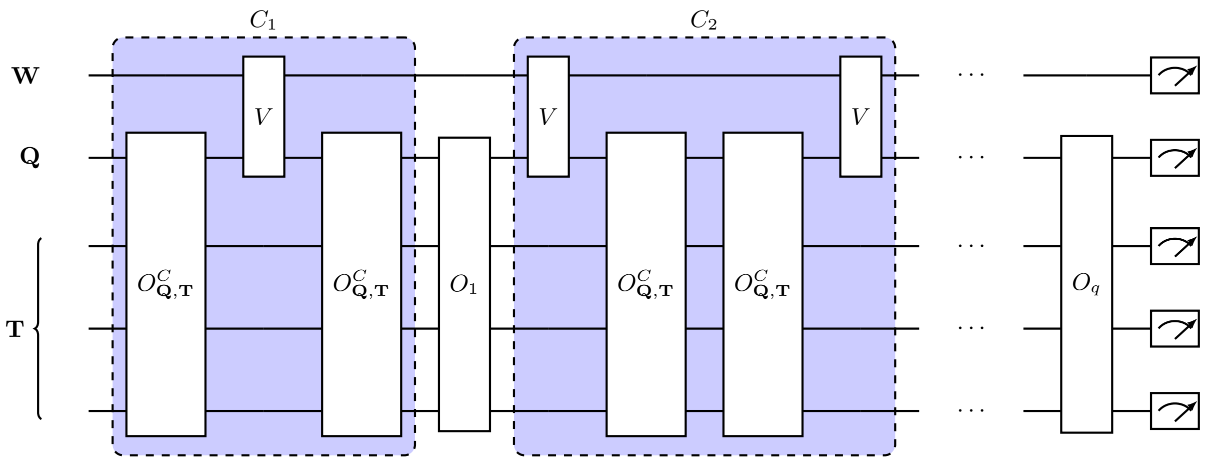

Figure 2: The behavior of the quantum subroutine: $O_1, ..., O_q$ denote the unitary operation that includes a single quantum group operation, and $C_0, ..., C_{q-1}$ denote quantum algorithms that may include multiple classical group operations but no quantum group operations. V denotes an arbitrary quantum algorithm. All registers are measured after O_q on a computational basis.

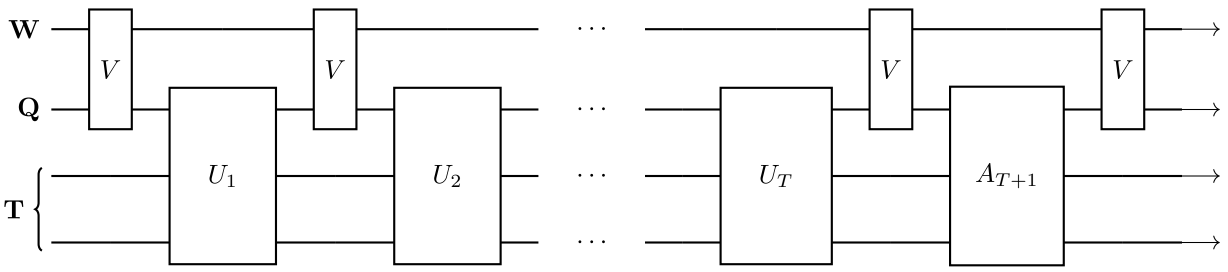

Figure 3: The generic hybrid algorithm with T invocations of quantum subroutines: $U_1, ..., U_T$ are quantum subroutines and include the measurement at the end. A_{T+1} is a generic algorithm with classical group operations. V denotes an arbitrary quantum algorithm.

parallel quantum group operation. The subroutine is forced to measure its all registers after O_d.

We characterize a parallel-query generic hybrid algorithm by a sequence of generic algorithms $(U_1, ..., U_T, A_{T+1})$ where U_j is a parallel-query generic quantum algorithm and A_{T+1} is a generic algorithm with only classical group operations.

Remark 7. We assume that the algorithm a priori fixes the sequence of oracle queries to classical and quantum group operations. This means that the algorithm cannot decide which oracle to call depending on its (classical) memory. As shown in [DFH22], our result holds for the adaptive algorithm that chooses which oracle to query based on its memory with a slightly worse bound.

Remark 8. This model of hybrid algorithms embraces a large class of hybrid algorithms considered in the literature as long as the subroutines have a bounded depth or number of quantum queries. For example, both d-CQ and d-QC schemes in [CCL23] are included in our model. Even the algorithms in a higher hierarchy like $\mathsf{BPP}^{\mathsf{BQNC}^{\mathsf{BPP}}}$ (which is advocated in [ACC+22] as a proper model of hybrid algorithms) are described in this model, provided that the query number/depth bounds hold. In particular, the d-depth quantum subroutine can be interpreted as a d-QC scheme.

5.2 Lower Bounds for Hybrid Algorithms

This section presents two types of quantum query lower bounds of the DL problem against hybrid algorithms: The generic hybrid algorithm with q-query and d-depth quantum subroutines. We begin with the following simulation theorem for hybrid algorithms, whose proof is deferred to the end of this section.

Theorem 5.1. *Let $\mathcal{G}$ be a group. Suppose that a generic hybrid algorithm $A^{\mathcal{G}}$, taking m group elements as inputs, makes at most Q group operation queries (including both classical and quantum).*

- *If $A^{\mathcal{G}}$ invokes q-query quantum subroutines T times, then there exists a (randomized) classical GGM algorithm $B^{\mathcal{G}}$ that perfectly simulates $A^{\mathcal{G}}$ with*
$$Q + T \cdot 2^{q+1}(m+Q+1)^{2q}$$
classical queries.

- *If $A^{\mathcal{G}}$ invokes d-depth quantum subroutines T times, then there exists a (randomized) classical GGM algorithm $B^{\mathcal{G}}$ that perfectly simulates $A^{\mathcal{G}}$ with*
$$Q + T \cdot 2^{2^d}(m+Q+1)^{2^d}$$
classical queries.

As corollaries, we prove the lower bounds of the hybrid algorithm.

Theorem 5.2. *Let $\mathcal{G}$ be a prime-order cyclic group. Any constant-advantage generic hybrid algorithm solving the DL/CDH/DDH problems with $Q = O(\operatorname{poly}\log|\mathcal{G}|)$ group operations, including both classical and quantum, must make $\Omega(\log|\mathcal{G}|/\log\log|\mathcal{G}|)$ quantum queries of depth $\Omega(\log\log|\mathcal{G}|)$ between some two consecutive forced measurements.*

More precisely, the following holds. Let $q \geq 1, d \geq 0$ be a positive integer. Let $* \in \{\mathsf{DL}, \mathsf{CDH}, \mathsf{DDH}\}$.

- *If a generic hybrid algorithm $A^{\mathcal{G}}$ invokes q-query quantum subroutines T times for $q \geq 1$, then it holds that*
$$\mathsf{Adv}_*(A^{\mathcal{G}}) = O\left(\frac{\left(T \cdot 2^{q+1}(m+Q+1)^{2q}\right)^2}{|\mathcal{G}|}\right).$$

- *If a generic hybrid algorithm $A^{\mathcal{G}}$ invokes d-depth quantum subroutines T times, then it holds that*
$$\mathsf{Adv}_*(A^{\mathcal{G}}) = O\left(\frac{\left(T \cdot 2^{2^d}(m+Q+1)^{2^d}\right)^2}{|\mathcal{G}|}\right).$$

5.3 Proof of the Hybrid Simulation Theorem

This section proves Theorem 5.1. We first show the following variants of Theorem 4.1 simulating the quantum subroutines.

Lemma 5.3 (The simulation of q-query parallel quantum subroutine)**.** *Let $\mathcal{G}$ be a group. Suppose that $A^{\mathcal{G}}$ is a q-query quantum subroutine in the QGGM, taking m group elements and a classical string as inputs, makes at most Q group operation queries including both classical and quantum. Then there exists a generic algorithm $B^{\mathcal{G}}$ in the GGM with*

$$Q + 2^{q+1}(m+Q+1)^{2q}$$

classical queries such that the output distribution of $B^{\mathcal{G}}(y)$ and $A^{\mathcal{G}}(y)$ are identical for any input y.

Lemma 5.4 (The simulation of d-depth parallel quantum subroutine)**.** *Let $\mathcal{G}$ be a group. Suppose that $A^{\mathcal{G}}$ is a d-depth parallel quantum subroutine in the QGGM, taking m group elements and a classical string as inputs, makes at most Q classical group operation queries. Then there exists a generic algorithm B in the GGM with*

$$2^{2^d}(m+Q+1)^{2^d}$$

classical queries such that the output distribution of $B^{\mathcal{G}}(y)$ and $A^{\mathcal{G}}(y)$ are identical for any input y.

We can prove Theorem 5.1 by invoking the above lemmas for each quantum subroutine.

Proof of Theorem 5.1. Let A be a generic hybrid algorithm in the QGGM with T invocations of quantum subroutines. Suppose that A is characterized by $(U_1, ..., U_T, A_{T+1})$ where U_j is a q-query quantum subroutine (or d-depth quantum subroutine), and A_{T+1} is a generic algorithm in the QGGM with classical group operations only. Let c be the number of queries A_{T+1} made and Q_j the total number of group operations U_j made.

We construct an algorithm B in the GGM that perfectly simulates A. We use the fact that the table register of the output of a quantum subroutine with m group elements as input and Q total queries has at most $m+Q$ nonzero group elements as each group operation makes at most a single new group element.

As the first step, B simulates U_1 using a sub-algorithm B_1. Lemmas 5.3 and 5.4 assert that B_1 can simulate the q-query and d-depth quantum subroutine with

$$Q_1 + 2^{q+1}(m+Q_1+1)^{2q}, \text{ and } 2^{2^d}(m+Q_1+1)^{2^d}$$

queries, respectively.

After the simulation, B discards all but the output of the simulation, which is safely done as all registers are measured on a computational basis at the end of execution of U_1. Below, we only consider the non-discarded parts when we say the group elements or table register, etc. The discarded parts do not affect the remaining simulation.

The table register has at most $m + Q_1$ nonzero group elements after U_1 since Q_1 queries of U_1 add at most Q_1 new group elements to the table. B then simulates U_2 using the non-discarded parts as input[17] with a new sub-algorithm B_2. The complexity of the simulation is similar, and the result of measurement gives at most $m + Q_1 + Q_2$ nonzero group elements in the table.

Continuing this procedure until U_T. Finally, B simulates A_{T+1}. Each classical group operation query of A_{T+1} can be simulated by a single group operation of B, because the table register is

[17]Technically, we need slight variants of the above lemmas for the simulation, as the simulation input is described in the GGM, while the subroutines in the lemmas expect the group elements stored in the QGGM. Still, the procedures are identical, and we choose the modular analysis and omit the subtle details.

measured after the execution of each quantum subroutine, and it is always classical during the execution of A_{T+1}. In other words, the simulation of A_{T+1} requires c classical group operations.

We conclude that the algorithm B in the GGM simulates the algorithm A in the QGGM. The number of group operations is, if A invokes q-query quantum subroutines,

$$\left(Q_1 + 2^{q+1}(m + Q_1 + 1)^{2q}\right) + \left(Q_2 + 2^{q+1}(m + Q_1 + Q_2 + 1)^{2q}\right) + ... + c \le Q + T \cdot 2^{q+1}(m+Q+1)^{2q},$$

and

$$2^{2^d}(m + Q_1 + 1)^{2^d} + 2^{2^d}(m + Q_1 + Q_2 + 1)^{2^d} + ... + c \le Q + T \cdot 2^{2^d}(m + Q + 1)^{2^d}$$

if A invokes d-depth quantum subroutines, where we use $Q_1 + Q_2 + ... + Q_T + c \le Q$. $\square$

It remains to prove Lemmas 5.3 and 5.4. We first prove the d-depth parallel quantum subroutine case, which can be proven similarly to Theorem 4.1 with some modifications. The query number case needs a new idea considering the branches and the modifications used in the depth case.

5.3.1 Proof of Lemma 5.4

The basic idea of the proof is that, when A makes a classical group operation, B can update the label function L and the set S with a single group operation. This is because the algorithms measure the relevant registers. As many parts of the proofs resemble one of Theorem 4.1, we highlighted the differences in the simulation in red.

Proof of Lemma 5.4. Let A be a generic quantum subroutine in the QGGM with at most d quantum query depth that takes m group elements $y_1, ..., y_m$ and classical string z as input. We assume that $y_1, ..., y_m$ are not 0, and let $N = |\mathcal{G}|$. Suppose that A is characterized by a sequence of algorithms $C_0, O_1, ..., C_{d-1}, O_d$ with generic algorithm C_j with c_j classical queries and generic algorithm O_j with a single parallel quantum group operation.

We construct a generic algorithm B in the GGM that simulates A by following the construction in Theorem 4.1, except for some specifications of group operation queries. In particular, the local operation, the equality query, and the finalization step are identical. The initialization is only slightly different as the algorithm A takes a classical string as a part of the input. We also apply the parallel group operation query procedure for the parallel quantum group operation queries, but we need a slightly different procedure for classical group operations. The correctness of the simulation procedure can be proven in the same way as the original proof.

Initialization. The algorithm B parses the input into the classical string z and group elements $y_1, ..., y_m$ stored in the table register, and prepares a label function $L : \mathbb{Z}_N[Y_1, ..., Y_m] \to [N] \cup \{\perp\}$ and $S \subseteq \mathbb{Z}_N[Y_1, ..., Y_m]$. Set $S := \{0\} \cup \{Y_i\}_{i \in \{1,...,m\}}$ and initialize L as in the previous proof. B creates the following state as a simulation of the initial state for A:

$$|z\rangle_{\mathbf{W},\mathbf{Q}} \otimes |L(Y_1), ..., L(Y_m), L(0), ..., L(0)\rangle_{\mathbf{T}}$$

Recall that $S \subset \mathbf{Z}_N[Y_1, ..., Y_m]$ is the set of polynomials on which the value of L is defined. Before describing the classical group operation simulation, we make the following observation for the parallel quantum group operation query.

Claim 1. Suppose that S_{pre} be a set S right before a parallel quantum group operation and S_{post} be a set S right after the same parallel quantum group operation. Then it holds that $|S_{\mathsf{post}}| \leq 2|S_{\mathsf{pre}}|^2$.

Proof. In the parallel group operation query simulation, the set of new elements added in S_{post} is included in the set

$$S' := \{h = f \pm g : f, g \in S_{\mathsf{pre}}\},$$

which has at most $2|S_{\mathsf{pre}}|^2$ different elements. As S_{pre} includes 0 by definition, $S_{\mathsf{pre}} \subset S'$ holds. This implies the claim. $\square$

Classical group operation query. When A makes a classical group operation query, B does the following. Let $S_{\mathsf{pre}} := S$.

1. Measure B's query register $\mathbf{Q}$ to obtain (b, i, j), and do nothing if $i = j$. Otherwise, measure the i-th and j-th entries of B's table register to obtain the labels ℓ_i, ℓ_j.
2. Find a pair $(f, g) \in S_{\mathsf{pre}}^2$ such that $\ell_i = L(f)$ and $\ell_j = L(g)$. Check if there is any $h \in S$ such that
$$h(y_1, ..., y_m) = f(y_1, ..., y_m) + (-1)^b g(y_1, ..., y_m).$$
   - If there is such h, it sets $L(f + (-1)^b g) := L(h)$.
   - Otherwise, it uniformly sets $L(f + (-1)^b g) \leftarrow [N]$ under the constraint that $L(f + (-1)^b g) \neq L(h)$ for all $h \in S$.

   Then update $S \leftarrow S \cup \{f + (-1)^b g\}$.
3. For all pairs $(f', g') \in S_{\mathsf{pre}}^2$ such that $L(f') = \ell_i$ and $L(g') = \ell_j$, set $L(f' + (-1)^b g') := L(f + (-1)^b g)$ and update $S \leftarrow S \cup \{f' + (-1)^b g'\}$.
4. Define $\ell_i + (-1)^b \ell_j := L(f + (-1)^b g)$.
5. Then, B simulates the classical group operation oracle as follows:
$$|b, i, j\rangle_{\mathbf{Q}} \otimes |..., \ell_i, ..., \ell_j, ...\rangle_{\mathbf{T}} \mapsto |b, i, j\rangle_{\mathbf{Q}} \otimes \left|..., \ell_i + (-1)^b \ell_j, ..., \ell_j, ...\right\rangle_{\mathbf{T}}$$
   for $i \neq j$, and otherwise it does nothing.

Since the register $\mathbf{Q}$ and the i-th and j-th entries of $\mathbf{T}$ are measured, this step only needs a single group operation for computing $f(y_1, ..., y_m) + (-1)^b g(y_1, ..., y_m)$.

Query complexity. We count the number of classical group operations made by B. Let S_{j-1} be the set S right before the parallel quantum group operation in O_j and let S_d be the final set S. Recall C_0 makes c_0 classical group operations and the simulation of each classical group operation takes a single group operation by B. We have $|S_0| \leq m + 1 + c_0$, where $m + 1$ is the initial elements included in S. Also, using Claim 1 and the fact that C_j makes c_j classical group operations, we have

$$|S_j| \leq 2|S_{j-1}|^2 + c_j,$$

and $|S_d| \leq 2|S_{d-1}|^2$. Since $|S_d|$ is an upper bound of the number of queries, we conclude that B can perfectly simulate A with

$$2^{2^d}(m+1+c_0+c_1+...+c_{d-1})^{2^d} \leq 2^{2^d}(m+Q+1)^{2^d}$$

group operations. This concludes the result for the d-depth parallel quantum subroutine. $\square$

5.3.2 Proof of Lemma 5.3

The main idea of this case is to consider the branches. In the beginning, there is only a single branch with nonzero amplitude in the table register. Two observations for proving Lemma 5.3 are 1) for simulating a group operation over a fixed (classical) table in a single branch, we only need a tiny number of new group elements, and 2) for each group operation, the number of branches is multiplied by a bounded number; looking ahead, it is $2(m+Q+1)^2$. As the simulation procedure in Theorem 4.1 does not consider the branch, we need some more work in this case.

Proof of Lemma 5.3. Let A be a q-query quantum subroutine characterized by $(C_0, O_1, ..., C_{q-1}, O_q)$. Here, C_j is a generic algorithm with c_j classical group operations and O_j is a generic algorithm with a single quantum group operation. We assume that A takes m group elements $y_1, ..., y_m$ and classical string z as input and suppose that each y_j is not 0, and let $N = |\mathcal{G}|$. As in the above proof, we construct a generic algorithm B in the GGM that simulates A.

Initialization. The algorithm B parses the input into the classical string z and group elements $y_1, ..., y_m$ stored in the table register. B prepares a label function $L : \mathbb{Z}_N[Y_1, ..., Y_m] \to [N] \cup \{\perp\}$ and $S \subseteq \mathbb{Z}_N[Y_1, ..., Y_m]$. Set $S := \{0\} \cup \{Y_i\}_{i \in \{1,...,m\}}$ and initialize L as in the previous proof. B creates the following state as a simulation of the initial state for A:

$$|z\rangle_{\mathbf{W},\mathbf{Q}} \otimes |L(Y_1), ..., L(Y_m), L(0), ..., L(0)\rangle_{\mathbf{T}}$$

Additionally, B initializes a rooted tree structure $\mathcal{T}$ with a root $v_0 = \{L(0)\} \cup \{L(Y_i)\}_{i \in \{1,...,m\}}$, without any further vertex. During the simulation, the tree $\mathcal{T}$ will be updated along with S and L, so that a leaf node represents the table register of a branch with a nonzero amplitude. For example, the unique leaf node v_0 of the initial tree $\mathcal{T}$ includes all information of the table register of the unique initial branch.

Quantum group operation query. Suppose A makes a quantum group operation query. Let $S_{\mathsf{pre}} := S$ and $\mathcal{T}_{\mathsf{pre}} := \mathcal{T}$ for recording the set S and $\mathcal{T}$ at the point of making the query. Then B does the following:

1. For each leaf node v_l of $\mathcal{T}_{\mathsf{pre}}$ (in arbitrary order), and for each $(\ell, \ell') \in v_l^2$, do the following:
   (a) Find a pair $(f, g) \in S_{\mathsf{pre}}^2$ such that $L(f) = \ell$ and $L(g) = \ell'$. Check if there is any $h \in S$ such that
   $$h(y_1, ..., y_m) = f(y_1, ..., y_m) + g(y_1, ..., y_m).$$
   - If there exists such h, it sets
   $$L(f+g) := L(h).$$

        - Otherwise, it uniformly sets $L(f+g) \leftarrow [N]$ under the constraint that $L(f+g) \neq L(h)$ for all $h \in S$.

   (b) Update $S \leftarrow S \cup \{f+g\}$. Add $v_{l+1} := v_l \cup \{L(f+g)\}$ as a child node of v_l.

   (c) For all pairs $(f', g') \in S_{\mathsf{pre}}^2$ such that $L(f') = \ell$ and $L(g') = \ell'$, set $L(f'+g') := L(f+g)$ and update $S \leftarrow S \cup \{f'+g'\}$.

2. Similarly add $v_{l+1} := v_l \cup \{L(f-g)\}$ as a child node of v_l and update S and L properly.

3. Define $\ell \pm \ell'$ for labels $(\ell, \ell') \in [N]^2$ as follows:

   - Check if there is $(f, g) \in S_{\mathsf{pre}}^2$ such that $\ell = L(f)$ and $\ell' = L(g)$.
     – If they exist, define
     $$\ell \pm \ell' := L(f \pm g).$$
     – Otherwise, define $\ell \pm \ell' := \perp$.

4. Apply the unitary $\widetilde{O}_{\mathbf{Q},\mathbf{T}}$ that is defined by:
$$|b, i, j\rangle_{\mathbf{Q}} \otimes |..., \ell_i, ..., \ell_j, ...\rangle_{\mathbf{T}} \mapsto |b, i, j\rangle_{\mathbf{Q}} \otimes \left|..., \ell_i + (-1)^b \ell_j, ..., \ell_j, ...\right\rangle_{\mathbf{T}}$$
   if $i \neq j$ and otherwise it does nothing.

Classical group operation query. When A makes a classical group operation query, B does the following. This is identical to the classical group operation in the d-depth subroutine case, except for the update of $\mathcal{T}$. Precisely, in the second step, B updates $v_l \leftarrow v_l \cup \{L(f + (-1)^b g)\}$ for each leaf node v_l.

Finalization. Finalization is identical to the previous simulations.

Analysis. The main difference of this simulation from the previous simulations is using $\mathcal{T}$. In particular, B computes the labels for $f \pm g$ only when f, g comes from a single leaf node. We argue that it suffices for simulating the group operation queries. For an intermediate quantum state
$$\sum_{w,b,i,j,X} \alpha_{w,b,i,j,X} |w\rangle_{\mathbf{W}} |b, i, j\rangle_{\mathbf{Q}} |X\rangle_{\mathbf{T}},$$
we say that X is a *nontrivial* table if $\alpha_{w,b,i,j,X}$ is nonzero for some w, b, i, j.

Claim 2. Right before the group operation query, for a nontrivial table X, there is a leaf node v_l of $\mathcal{T}$ such that the following holds: For any group element included in X, there is $f \in v_l$ such that $f(y_1, ..., y_m) = x$. We say that v_l corresponds to X.

Proof. We use induction. Before the first group operation query, the unique nontrivial table $X = (y_1, y_2, ..., y_m, 0, ...)$ corresponds to the initial leaf node $v_0 = \{L(0)\} \cup \{L(Y_i)\}_{i \in \{1,...,m\}}$. Consider a group operation query, in which the statement holds right before the query. We prove that the statement still holds after the query.

Suppose that the group operation is quantum. Let $X = (x_1, x_2, ...)$ be a nontrivial table, and let v_l be the corresponding leaf before the query at this point. Consider a fixed branch

$$|w\rangle_{\mathbf{W}} |b, i, j\rangle_{\mathbf{Q}} |X\rangle_{\mathbf{T}}.$$

Since v_l corresponds to X, there exist $f, g \in S$ such that $f(y_1, ..., y_m) = x_i$, $g(y_1, ..., y_m) = x_j$ and $L(f), L(g) \in v_l$ hold. The simulation appends $v_{l+1} = v_l \cup \{L(f + (-1)^b g)\}$ as a child node of v_l. Since the quantum group operation query only changes the i-th entry by x_i to $x_i + (-1)^b x_j$, we can easily check the v_{l+1} corresponds to the table after query.

For a classical group operation query, a similar argument works with the same path v_l since the simulation adds $L(f + (-1)^b g)$ to the set v_l. □

This claim ensures that each nontrivial table corresponds to some leaf node in the simulation so that the simulation works well as in the previous proofs.

In the remainder of the proof, we count the number of group operations made by B. To do so, we need some calculations for leaf nodes. Let v_l be a leaf node. B appends the child nodes having one more group element that v_l for each quantum group operation. Similarly, B adds a single group element to the leaf nodes for each classical group operation. This implies that all nodes have at most $m + 1 + Q$ elements during the simulation.

After the initialization, there is only a single leaf node. For each quantum group operation, B appends at most $2|v_l|^2$ child nodes to the leaf node v_l. Using $|v_l| \leq m + Q + 1$, the final tree $\mathcal{T}$, after q quantum group operations, has at most

$$2^q (m + Q + 1)^{2q}$$

leaf nodes. Since each non-root node and each classical group operation requires a single group operation to update, the total number of queries made by B is bounded by

$$Q + 2^q (m + Q + 1)^{2q} + 2^{q-1} (m + Q + 1)^{2(q-1)} + ... \leq Q + 2^{q+1} (m + Q + 1)^{2q}.$$

This concludes the proof. □

6 Memory-bounded Algorithms

In this section, we consider the generic algorithms with *bounded memory*. More precisely, we consider algorithms that have two memories, classical and quantum, and that store a limited number of group elements in their quantum register. Furthermore, we assume that the algorithms can coherently access a few group elements stored in the classical memory in a single group operation. In other words, we assume that the algorithms only have a small amount of quantum random access classical memory (qRAM).

6.1 Quantum and Classical Memory Models

Recall that algorithms in (Q)GGM interact with a black-box table register $\mathbf{T}$ that stores group elements in $\mathbb{Z}_N$. In this section, we assume that $\mathbf{T}$ holds two components $\mathbf{T}_C$ and $\mathbf{T}_Q = \mathcal{H}_{\mathbb{Z}_N}^{\otimes t}$. The second component, $\mathbf{T}_Q$, is a quantum memory with a bounded size. Generic algorithms can coherently access or store group elements in $\mathbf{T}_Q$ in superposition.

The first component $\mathbf{T}_C$, corresponding to the classical memory, always stores group elements in a computational basis, i.e., classical group elements. We additionally assume that the quantum algorithm has a restriction on *coherently* accessing $\mathbf{T}_C$. In other words, $\mathbf{T}_C$ is not a quantum random access (classical) memory (qRAM). We may assume that there is a qRAM holding a small number of group elements, which will be formally discussed below.

Recall that the register $\mathbf{T}$ is of the form $\mathbf{T} = \mathcal{H}_{\mathbb{Z}_N}^{\otimes s}$. We decompose it into $\mathbf{T}_Q \otimes \mathbf{T}_C$ for $\mathbf{T}_Q = \mathcal{H}_{\mathbb{Z}_N}^{\otimes t}$ where $s \geq t$. We additionally assume that each i-th component of $\mathbf{T}$ for $i > t$ always holds a classical group element. The quantum group operation query

$$|b, i, j\rangle_{\mathbf{Q}} \otimes |..., x_i, ..., x_j, ...\rangle_{\mathbf{T}} \mapsto |b, i, j\rangle_{\mathbf{Q}} \otimes \left|..., x_i + (-1)^b x_j, ..., x_j, ...\right\rangle_{\mathbf{T}}$$

has the following restrictions: In each query, the indices should be one of the following choices and obey the corresponding conditions.

1. (Group operations for quantum registers) The second and third registers of $\mathbf{Q}$ hold indices (that may be in superpositions) indicating the group elements in the quantum register $\mathbf{T}_Q$; that is, $i, j \leq t$ (in any branch) always holds.

2. (Group operations for quantum-classical registers) The second register of $\mathbf{Q}$ holds indices (that may be in a superposition) indicating the group elements in $\mathbf{T}_Q$, and the third register of $\mathbf{Q}$ is classical (i.e., measured before query) and indicates a group element in $\mathbf{T}_C$; that is, $i \leq t$ (in any branch) and $j > t$ holds.

3. (Group operations for classical registers) The second and third registers of $\mathbf{Q}$ are both classical (i.e., measured in the computational basis before query), and the stored indices i, j indicate group elements in $\mathbf{T}_C$; that is, $i, j > t$ holds.

The first and second options are basically quantum (if the algorithm does not measure $\mathbf{Q}$), and the last option should be classical. Therefore, combining the first and second group operations as quantum group operations is convenient.

In general, we consider the case that a (small) qRAM is available. We model a qRAM as a storage containing r group elements. The qRAM is only involved in quantum-classical group operations. When querying a quantum group operation, the algorithm must specify r indices $j_1, ..., j_r > t$ for the classical register. The qRAM loads those elements, and the group operation is made between qRAM and quantum register $\mathbf{T}_Q$ or just in $\mathbf{T}_Q$, after which the data in qRAM is discarded. Note that the second register must hold indices (in a superposition) indicating the group elements in $\mathbf{T}_Q$ since the result of the quantum group operation is written in that register. Therefore, the quantum group operation can be described as follows, by omitting the qRAM's data loading and deletion.

- (Group operations for quantum registers and qRAM) The algorithm specifies (not in superposition) a set $J = \{j_1, ..., j_r\}$ of indices such that $j_k > t$ for all $k = 1, ..., r$. The second register must indicate the group elements in $\mathbf{T}_Q$, possibly in a superposition. The third register holds indices $j_1, ..., j_r$ or indicates the group elements in $\mathbf{T}_Q$. Then it applies $O_{\mathbf{Q},\mathbf{T}}$.

In summary, this section deals with the generic hybrid quantum-classical algorithm in the QGGM with q-query quantum subroutines as described in Section 5, with the additional memory constraints described above. The group operations for classical registers (Item 3) are always considered as classical group operations. The quantum group operations are always the above group operations for quantum registers and qRAM.

6.2 Lower Bounds for Memory-Bounded Algorithms

We present the simulation theorem for memory-bounded algorithms first.

Theorem 6.1. *Let $\mathcal{G}$ be a group. Suppose that a generic hybrid algorithm $A^{\mathcal{G}}$ makes C classical queries and invokes q-query quantum subroutines T times. If the quantum memory of $A^{\mathcal{G}}$ only can store t group elements and $A^{\mathcal{G}}$ can access qRAM of r group elements, then there exists a (randomized) classical GGM algorithm $B^{\mathcal{G}}$ that perfectly simulates $A^{\mathcal{G}}$ with*

$$C + T \cdot 2^{q+1}(t \cdot (t-1+r))^q$$

classical group operations.

Similarly to the other cases, this theorem directly implies the following lower bounds.

Theorem 6.2. *Let $\mathcal{G}$ be a prime-order cyclic group. Any constant-advantage generic hybrid algorithm solving the DL/CDH/DDH problems with quantum memory holding $t = O(1)$ group elements and qRAM of $r = O(1)$ group elements must make either $C = \Omega(\sqrt{|\mathcal{G}|})$ classical group operations or $\Omega(\log|\mathcal{G}|)$ quantum group operations.*

More precisely, the following holds. If a generic hybrid algorithm $A^{\mathcal{G}}$ in the QGGM invokes q-query quantum subroutines T times, it holds that for any $* \in \{\mathsf{DL}, \mathsf{CDH}, \mathsf{DDH}\}$

$$\mathsf{Adv}_*(A^{\mathcal{G}}) = O\left(\frac{\left(C + T \cdot 2^{q+1}(t \cdot (t-1+r))^q\right)^2}{|\mathcal{G}|}\right).$$

6.3 Proof of the Memory-Bounded Simulation Theorem

We prove the following memory-bounded simulation theorem for the quantum subroutines. The proof of Theorem 6.1 is almost identical to the hybrid case, except that it uses Lemma 6.3 that has a nice property that the number of classical queries C is not involved in the exponential term.

In the proof of Lemma 6.3, the main difference is the contents of $\mathcal{T}$ where each node only includes t elements, and identifies a potential branch in $\mathbf{T}_Q$.

Lemma 6.3. *Let $\mathcal{G}$ be a group. Suppose that $A^{\mathcal{G}}$ is a q-query quantum subroutine in the QGGM, taking group elements and a classical string as inputs, makes at most C classical group operation. If the quantum memory of $A^{\mathcal{G}}$ only can store t group elements and $A^{\mathcal{G}}$ can access qRAM of r group elements, then there exists a generic algorithm $B^{\mathcal{G}}$ in the GGM with*

$$C + 2^{q+1}(t(t-1+r))^q$$

classical queries such that the output distribution of $B^{\mathcal{G}}(y)$ and $A^{\mathcal{G}}(y)$ are identical for any input y.

Proof. The proof is almost identical to that of Lemma 5.3, except that we change the contents of the rooted tree structure to only include group elements in quantum memory.

Let A be a q-query quantum subroutine that is characterized by $(C_0, O_1, ..., C_{q-1}, O_q)$. Here, C_j and O_j are generic algorithms with c_j classical group operations and a single quantum group operation, respectively. We assume that A takes m group elements $y_1, ..., y_m$ classical string z as input. Suppose that the input group elements are not 0, and let $N = |\mathcal{G}|$. We will construct a generic algorithm B in the GGM that simulates A.

Initialization. The algorithm B prepares a label function $L : \mathbb{Z}_N[Y_1, ..., Y_m] \to [N] \cup \{\perp\}$ and $S \subseteq \mathbb{Z}_N[Y_1, ..., Y_m]$. Set $S := \{0\} \cup \{Y_i\}_{i\in\{1,...,m\}}$ and initialize L as in the previous proof. B creates the following state as a simulation of the initial state for A:

$$|z\rangle_{\mathbf{W},\mathbf{Q}} \otimes |L(Y_1), ..., L(Y_m), L(0), ..., L(0)\rangle_{\mathbf{T}}$$

Additionally, B initializes a rooted tree structure $\mathcal{T}$ with a root $v_0 = \{L(Y_i)\}_{i\in\{1,...,t\}}$ *as a multiset*, without any further vertex. The nodes of $\mathcal{T}$ indicate potential branches of quantum memory $\mathbf{T}_Q$. If a branch includes a group element multiple times, the corresponding multiset will include the same element with the same multiplicity. During the simulation, the tree $\mathcal{T}$ will be updated along with S and L.

Local operation. When A applies some operation on its local registers $\mathbf{W}, \mathbf{Q}$, B also applies the same operation.

Quantum group operation query. Suppose A makes a quantum group operation query. Note that A specifies the qRAM index set $J = \{j_1, j_2, ..., j_r\}$, which can be obtained by B as well. Let $S_{\mathsf{pre}} := S$ and $\mathcal{T}_{\mathsf{pre}} := \mathcal{T}$ for recording the set S and $\mathcal{T}$ at the point. Then B does the following:

1. For each leaf node v_l of $\mathcal{T}_{\mathsf{pre}}$ (in arbitrary order), do the following:
   (a) For each $(\ell, \ell') \in v_l^2$, do the following:
      i. Find a pair $(f, g) \in S_{\mathsf{pre}}^2$ such that $L(f) = \ell$ and $L(g) = \ell'$. Check if there is any $h \in S$ such that
      $$h(y_1, ..., y_m) = f(y_1, ..., y_m) + g(y_1, ..., y_m).$$
         - If there exists such h, it sets
         $$L(f + g) := L(h).$$
         - Otherwise, it uniformly sets $L(f+g) \leftarrow [N]$ under the constraint that $L(f+g) \neq L(h)$ for all $h \in S$.

      ii. Update $S \leftarrow S \cup \{f + g\}$. Define $v_{l+1} := v_l \setminus \{L(f)\} \cup \{L(f + g)\}$.[18] Add v_{l+1} as a child node of v_l.
      iii. For all pairs $(f', g') \in S_{\mathsf{pre}}^2$ such that $L(f') = \ell$ and $L(g') = \ell'$, set $L(f' + g') := L(f + g)$ and update $S \leftarrow S \cup \{f' + g'\}$.
      iv. Similarly add $v_{l+1} := v_l \setminus \{L(f)\} \cup \{L(f - g)\}$ as a child node of v_l and update S and L properly.

   (b) For each $\ell \in v_l$ and $k \in [r]$, do the following:
      i. Obtain the label ℓ' from the j_k-th entry of B's table register, and find a pair $(f, g) \in S_{\mathsf{pre}}^2$ such that $\ell = L(f)$ and $\ell' = L(g)$. Note that the j_k-th entry of B is classical and B needs not to measure it.
      ii. Check if there is any $h \in S$ such that
      $$h(y_1, ..., y_m) = f(y_1, ..., y_m) + g(y_1, ..., y_m).$$

[18] If there is multiple $L(f)$ in v_l, it reduces the multiplicity of f by 1.

   - If there exists such h, it sets
$$L(f+g) := L(h).$$
   - Otherwise, it uniformly sets $L(f+g) \leftarrow [N]$ under the constraint that $L(f+g) \neq L(h)$ for all $h \in S$.

   iii. Update $S \leftarrow S \cup \{f+g\}$. Define $v_{l+1} := v_l \setminus \{L(f)\} \cup \{L(f+g)\}$. Add v_{l+1} as a child node of v_l.

   iv. For all pairs $(f', g') \in S_{\text{pre}}^2$ such that $L(f') = \ell$ and $L(g') = \ell'$, set $L(f'+g') := L(f+g)$ and update $S \leftarrow S \cup \{f'+g'\}$.

   v. Similarly add $v_{l+1} := v_l \setminus \{L(f)\} \cup \{L(f-g)\}$ as a child node of v_l and update S and L properly.

2. Define $\ell \pm \ell'$ for labels $(\ell, \ell') \in [N]^2$ as follows:
   - Check if there is $(f,g) \in S_{\text{pre}}^2$ such that $\ell = L(f)$ and $\ell' = L(g)$.
     - If they exist, define
$$\ell \pm \ell' := L(f \pm g).$$
     - Otherwise, define $\ell \pm \ell' := \perp$.

3. Apply the unitary $\widetilde{O}_{\mathbf{Q},\mathbf{T}}$ that is defined by:
$$|b,i,j\rangle_{\mathbf{Q}} \otimes |..., \ell_i, ..., \ell_j, ...\rangle_{\mathbf{T}} \mapsto |b,i,j\rangle_{\mathbf{Q}} \otimes \left| ..., \ell_i + (-1)^b \ell_j, ..., \ell_j, ... \right\rangle_{\mathbf{T}}$$
   if $i \neq j$ and otherwise it does nothing.

Classical group operation query. When A makes a classical group operation query, B does the following. Let $S_{\text{pre}} := S$ and $\mathcal{T}_{\text{pre}} := \mathcal{T}$ for recording the set S and $\mathcal{T}$ at the point.

1. Measure B's query register $\mathbf{Q}$ to obtain (b,i,j), and do nothing if $i = j$. Otherwise, measure the i-th and j-th entries of B's table register to obtain the labels ℓ_i, ℓ_j.

2. Find a pair $(f,g) \in S_{\text{pre}}^2$ such that $\ell_i = L(f)$ and $\ell_j = L(g)$. Check if there is any $h \in S$ such that
$$h(y_1, ..., y_m) = f(y_1, ..., y_m) + (-1)^b g(y_1, ..., y_m).$$
   - If there is such h, it sets $L(f + (-1)^b g) := L(h)$.
   - Otherwise, it uniformly sets $L(f + (-1)^b g) \leftarrow [N]$ under the constraint that $L(f + (-1)^b g) \neq L(h)$ for all $h \in S$.

   Then update $S \leftarrow S \cup \{f + (-1)^b g\}$. If $i \leq t$, for each leaf node v_l of $\mathcal{T}_{\text{pre}}$ (in arbitrary order) such that $L(f) \in v_l$, update $v_l \leftarrow v_l \setminus \{L(f)\} \cup \{L(f + (-1)^b g)\}$.

3. For all pairs $(f', g') \in S_{\text{pre}}^2$ such that $L(f') = \ell_i$ and $L(g') = \ell_j$, set $L(f' + (-1)^b g') := L(f + (-1)^b g)$ and update $S \leftarrow S \cup \{f' + (-1)^b g'\}$.

4. Define $\ell_i + (-1)^b \ell_j := L(f + (-1)^b g)$.

5. Then, B simulates the classical group operation oracle as follows:

$$|b,i,j\rangle_{\mathbf{Q}} \otimes |...,\ell_i,...,\ell_j,...\rangle_{\mathbf{T}} \mapsto |b,i,j\rangle_{\mathbf{Q}} \otimes \left|...,\ell_i+(-1)^b\ell_j,...,\ell_j,...\right\rangle_{\mathbf{T}}$$

for $i \neq j$, and otherwise it does nothing.

Finalization. Finalization is identical to the previous simulations.

Analysis. The contents in each node is the main difference from the proof of Lemma 5.3. Recall that for an intermediate quantum state

$$\sum_{w,b,i,j,X} \alpha_{w,b,i,j,X} |w\rangle_{\mathbf{W}} |b,i,j\rangle_{\mathbf{Q}} |X\rangle_{\mathbf{T}},$$

we say that X is a nontrivial table if $\alpha_{w,b,i,j,X}$ is nonzero. We prove the following variant of Claim 2, which ensures that the algorithm B correctly simulates A.

Claim 3. Right before the group operation query, for a nontrivial table X, there is a leaf node v_l of $\mathcal{T}$ such that the following holds: Let $x_1,...,x_t$ be the first t elements of X. There is a leaf node $v_l = \{\ell_1,...,\ell_t\}$ in $\mathcal{T}$ such that there exist $(f_i)_{i\in[t]}$ where $f_i(y_1,...,y_m) = x_i$ and $L(f_i) = \ell_i$ for all $i \in [t]$. We say that v_l corresponds to X.

Proof. We use induction. Initially, the unique nontrivial table $X = (y_1,y_2,...,y_m,0,...)$ corresponds to the initial leaf node $v_0 = \{L(Y_1),...,L(Y_t)\}$. Consider a group operation query, in which the statement holds right before the query. We prove that the statement still holds after the query.

For the quantum group operation, let $X = (x_1,x_2,...)$ be a nontrivial table and $v_l = \{\ell_1,...,\ell_t\}$ be the corresponding leaf node. Let $(f_i)_{i\in[t]}$ be functions such that $f_i(y_1,...,y_m) = x_i$ and $L(f_i) = \ell_i$ for all $i \in [t]$. Consider a fixed branch

$$|w\rangle_{\mathbf{W}} |b,i,j\rangle_{\mathbf{Q}} |X\rangle_{\mathbf{T}}.$$

It holds that $f_i(Y_1,...,Y_m) = x_i$ for all $i \in [t]$. If $j > t$, let $f_j \in S$ be such that $f_j(Y_1,...,Y_m) = x_j$. After the query, it is easy to see that $v_{l+1} = v_l \setminus \{L(f_i)\} \cup \{L(f_i + (-1)^b f_j)\}$ defined above corresponds to the new table register. This argument also holds for the classical group operation with $i \leq t$. For the group operations over classical registers, v_l still corresponds to X. □

We count the number of group operations made by B. For each quantum group operation, B appends one child node when it makes one group operation for simulating A. On the other hand, each classical group operation of A requires a single group operation of B, and it may alter some contents of nodes but do not create any new node. Thus it suffices to count the number of nodes in $\mathcal{T}$ for computing the query complexity of B.

Recall that there are two options for quantum group operations. The following arguments show that each quantum query appends at most $2t(t-1+r)$ child nodes for each leaf node.

- For the operations for quantum registers, at most $2t(t-1)$ different branches can appear, where factor 2 represents the choice of $\pm$ and $t(t-1)$ is for the choice of indices in the quantum register.

- For the quantum-classical group operations with qRAM, the first index can be one of t elements in $\mathbf{T}_Q$ and the second index is one among $|J| = r$ group elements. Therefore, each query introduces at most $2tr$ new branches.

This implies that after q quantum queries, there are at most

$$1 + 2t(t-1+r) + (2t(t-1+r))^2 + \ldots + (2t(t-1+r))^q \leq 2^{q+1} (t(t-1+r))^q$$

different branches in $\mathcal{T}$ at the end, where we used $t, r \geq 1$. As a single classical group operation can be simulated by a single group operation of B, the total number of group operations in the GGM made by B is bounded by $C + 2^{q+1} (t(t-1+r))^q$, which concludes the proof. □

7 Quantum Algorithms in the QGGM

This section presents generic quantum algorithms for the DL and MDL problems. We first review Shor's algorithm for the DL problem with a closer look at the query complexity and its modification with classical preprocessing. The new MDL algorithm is presented at the end of this section.

Let N be a positive integer and define $w_N := \exp(2\pi i/N)$. The quantum Fourier transform QFT and its inverse QFT† are defined as follows:

$$\mathsf{QFT} : |x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} w_N^{xk} |k\rangle \text{ , and } \mathsf{QFT}^\dagger : |k\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} w_N^{-xk} |x\rangle .$$

7.1 The Discrete Logarithm Problem

The quantum algorithm for solving the DL problem follows the standard approach to the hidden subgroup problems (HSP). For completeness, we rephrase the quantum algorithm due to Shor [Sho94] in detail and describe its variations in the hybrid setting and the depth-efficient version. All of these algorithms are generic.

Let $\mathcal{G}$ be a cyclic group of order N. Suppose that a generator g and a handle g^x representing the problem instance are given to the adversary for random x. The QGGM algorithm below finds x, where the below description roughly includes the square-and-multiply method to compute g^{a+bx}, which are omitted in the usual descriptions.

7.1.1 The quantum DL algorithm

Let $|\mathcal{G}| = N$, $n = \lceil \log_2 N \rceil$. We set $s = 2n + 1$. The algorithm A proceeds as follows.

1. Given a problem instance (g, g^x), the algorithm prepares the group elements of the form g^{2^i} and $g^{2^j \cdot x}$, or the set

   $$D_x = \{g, g^2, g^{2^2}, \ldots, g^{2^{n-1}}\} \cup \{g^x, g^{2x}, g^{2^2 x}, \ldots, g^{2^{n-1}x}\}$$

   in the table using classical group operations. It prepares a quantum state $|0,0\rangle_A \otimes |1, D_x\rangle_T$.

2. Applying QFT $\otimes$ QFT on the working register of A to obtain

   $$\sum_{a,b=0}^{N-1} \frac{|a,b\rangle_A}{N} \otimes |1, D_x\rangle .$$

3. Using the binary expression of a, b and D_x, the algorithm computes

$$\sum_{a,b=0}^{N-1} \frac{|a,b\rangle_A \left|g^{a+bx}, D_x\right\rangle_T}{N},$$

applies $\mathsf{QFT}^\dagger \otimes \mathsf{QFT}^\dagger$, and measures the register A. For the measurement outcome is $(u,v) \neq (0,0)$, return v/u as an answer. Otherwise return $\perp$.

The number of oracle queries is $O(n)$; the construction of D_x requires $O(n)$ queries, and computing g^{a+bx} requires $2n$ queries, each of which is the controlled group operation multiplying g^{2^i} or $g^{2^j x}$ on the first entry of the table.

Note that the quantum registers of this algorithm are essentially the working register A and the first register of the table holding g^{a+bx}. Furthermore, the quantum group operation accesses only one register of the remaining parts. The following folklore theorem summarizes the result of this algorithm, regarding this observation.

Theorem 7.1. *Let $\mathcal{G}$ be a cyclic group. There exists a $O(\log|\mathcal{G}|)$-query QGGM algorithm that solves the discrete logarithm problem with an overwhelming probability.*

This algorithm requires a quantum register holding $2n$-qubit[19] *and a single group element, and classical storage holding $2n$ group elements. This algorithm does not require quantum access to classical storage.*[20]

7.1.2 Hybrid quantum-classical algorithms

In the above algorithm, the construction of D_x is entirely classical, and only the computation of g^{a+bx} uses quantum power. Since classical computation is much cheaper than quantum computing, it is tempting to reduce the later query complexity at the cost of the former classical preprocessing.

We can modify the above algorithm taking this consideration into account, by exploiting the base-p numeral system for $p \geq 2$. This modified p-base algorithm uses the following set

$$D_x^{(p)} = \bigcup_{1\leq k<p, 0\leq i<n_p} \left\{g^{k\cdot p^i}\right\} \cup \bigcup_{1\leq \ell<p, 0\leq j<n_p} \left\{g^{\ell\cdot p^j x}\right\}$$

instead of D_x above. The only differences are the step for preparing $D_x^{(p)}$ and the way to compute g^{a+bx}. The preparation requires $O(p\log|\mathcal{G}|/\log p)$ group operations, and the p-base exponent takes $O(\log|\mathcal{G}|/\log p)$ quantum group operations. It is worth noting that this algorithm requires the QRAM access to $D_x^{(p)}$, with the size p. The result of this hybrid algorithm is summarized as follows.

Theorem 7.2. *Let $\mathcal{G}$ be a cyclic group and let $p > 1$ be an integer. There exists a generic hybrid algorithm with $O(\log|\mathcal{G}|/\log p)$ quantum queries and $O(p\log|\mathcal{G}|/\log p)$ classical queries that solves the DL problem with an overwhelming probability.*

This algorithm requires a quantum register holding $2n$-qubit and a single group element, and classical storage holding $O(p\log|\mathcal{G}|/\log p)$ group elements. This algorithm requires QRAM access to the classical $O(p)$ group elements simultaneously.

[19] This can be reduced using the tricks in, e.g., [Kit96, ME98].

[20] Precisely speaking, it requires a qRAM storing a single group element, which is unavoidable when accessing group elements in classical memory.

7.1.3 Depth-efficient algorithms

As suggested in [CW00], we can exploit the binary tree with $2\log|\mathcal{G}|$ leaves and $\log 2\log|\mathcal{G}|$ depth to reduce the depth of the DL algorithm. Precisely, we prepare the elements in D_x as leaves of the binary tree and compute g^{a+bx} using this tree with $2\log|\mathcal{G}|-1$ internal nodes. Note that this computation is coherently done over internal nodes.

Furthermore, we can combine the binary tree idea with the base-p numeric system with $D_x^{(p)}$. In this case, we prepare $g^{(a_{2i}+a_{2i+1}\cdot p)p^{2i}}$ and $g^{(b_{2i}+b_{2i+1}\cdot p)p^{2i}}$ for indices a, b as leaf nodes by coherently multiplying the elements in $D_x^{(p)}$. The tree has $O(\log|\mathcal{G}|/\log p)$ nodes and $\log\log|\mathcal{G}| - \log\log p + O(1)$ depth. This gives the following depth-efficient DL algorithm.

Theorem 7.3. *Let $\mathcal{G}$ be a cyclic group and let p be an integer. There exists a generic quantum algorithm with $O(\log|\mathcal{G}|/\log p)$ queries and $\log\log|\mathcal{G}| - \log\log p + O(1)$ query depth that solves the discrete logarithm problem with an overwhelming probability. This algorithm requires a quantum register storing $2n$-qubit and $O(\log|\mathcal{G}|/\log p)$ group elements, and classical storage holding $O(p\log|\mathcal{G}|/\log p)$ group elements. This algorithm requires QRAM access to the classical $O(p)$ group elements simultaneously.*

7.2 The Multiple Discrete Logarithm Problem

This section describes our new quantum MDL algorithm, where the adversary is given m group elements $y_1 = g^{x_1}, ..., y_m = g^{x_m}$ as inputs. The adversary's goal is to find $x_1, ..., x_m$ using group operation queries.

The proposed algorithm follows the standard approach to hidden subgroup problems for the target function $f : \mathbb{Z}_N^{m+1} \to \mathcal{G}$ given by

$$f(k_0, ..., k_m) = g^{k_0} y_1^{k_1} \cdots y_m^{k_m}, \tag{2}$$

which hides a rank-m subgroup $H \le \mathbb{Z}_N^{m+1}$ generated by $\{x_i e_0 - e_i\}_{1\le i\le m}$; that is, it holds that $f(g_1) = f(g_2)$ for $g_1 H = g_2 H$. The improvement comes from the multi-exponentiation algorithm, which we recall below.

7.2.1 The multi-exponentiation problem

In the multi-exponentiation problem, we are given the elements $1, h_1, ..., h_m$ and the nonnegative exponents $e_1, ..., e_m$, and asked to find $h_1^{e_1} \cdot ... \cdot h_m^{e_m}$ only using the multiplication. Pippenger [Pip80] showed the following result, which is known to be almost optimal.

Proposition 7.4 ([Pip80]). *Let B be an integer, and $\lg m/\lg B = o(1)$. Suppose $e_i \le B$ for all i. Given inputs $1, h_1, ..., h_m$ and $e_1, ..., e_m$, there is an efficient deterministic algorithm to compute $h_1^{e_1} \cdot ... \cdot h_m^{e_m}$ with*

$$\lg B + \frac{(1+o(1))m\lg B}{\lg(m\lg B)}$$

multiplications.

7.2.2 A multiple discrete logarithm algorithm

Let $\mathcal{G}$ be a cyclic group with order N. Suppose that we are given $g, y_1 = g^{x_1}, ..., y_m = g^{x_m}$ as inputs. As in the standard algorithm for HSP, the algorithm prepares a superposition

$$\bigotimes_{i=0}^{m} \left(\sum_{0 \le k_i < N} \frac{|k_i\rangle}{\sqrt{M}} \right) = \sum_{0 \le k_0, ..., k_m < N} \frac{|k_0, ..., k_m\rangle}{\sqrt{M^N}}$$

using QFT and then compute the target function f in Equation (2) coherently using Proposition 7.4.

To execute Proposition 7.4, the condition $\lg m / \lg |\mathcal{G}| = o(1)$ must hold. We also note that this algorithm requires large quantum memory. The result in this section is as follows.

Theorem 7.5. *Let $\mathcal{G}$ be an cyclic group and m be a positive integer such that $\lg m / \lg |\mathcal{G}| = o(1)$. There exists a QGGM algorithm that solves the m-MDL problem using*

$$2 \log |\mathcal{G}| + \frac{(2 + o(1)) m \lg |\mathcal{G}|}{\lg(m \lg |\mathcal{G}|)}$$

quantum group operation with an overwhelming success probability. If $m = \Omega(\log |\mathcal{G}|)$, the amortized query complexity is $O(\log |\mathcal{G}| / \log m)$ per DL instance.

References

[Aar05] Scott Aaronson. Ten semi-grand challenges for quantum computing theory, July 2005. 8

[ACC⁺22] Atul Singh Arora, Andrea Coladangelo, Matthew Coudron, Alexandru Gheorghiu, Uttam Singh, and Hendrik Waldner. Quantum depth in the random oracle model. *arXiv preprint arXiv:2210.06454, to appear at STOC 2023*, 2022. 8, 21

[CCL23] Nai-Hui Chia, Kai-Min Chung, and Ching-Yi Lai. On the need for large quantum depth. *J. ACM*, 70(1):6:1–6:38, 2023. 8, 21

[CJS14] Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *J. Math. Cryptol.*, 8(1):1–29, 2014. 7

[CLM⁺18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018. 8

[CM20] Matthew Coudron and Sanketh Menda. Computations with greater quantum depth are strictly more powerful (relative to an oracle). In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Procceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 889–901. ACM, 2020. 8

[Cou06] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Paper 2006/291, 2006. https://eprint.iacr.org/2006/291. 8

[CW00] Richard Cleve and John Watrous. Fast parallel circuits for the quantum Fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 526–536. IEEE, 2000. 3, 6, 36

[DFH22] Jelle Don, Serge Fehr, and Yu-Hsuan Huang. Adaptive versus static multi-oracle algorithms, and quantum security of a split-key PRF. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I*, volume 13747 of *Lecture Notes in Computer Science*, pages 33–51. Springer, 2022. 21

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976. 3

[DHK$^+$23] Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. Cryptology ePrint Archive, Paper 2023/186, *to appear at PKC 2023*, 2023. https://eprint.iacr.org/2023/186. 8

[EH00] Mark Ettinger and Peter Høyer. On quantum algorithms for noncommutative hidden subgroups. *Adv. Appl. Math.*, 25(3):239–251, 2000. 7, 8

[EH17] Martin Ekerå and Johan Håstad. Quantum algorithms for computing short discrete logarithms and factoring RSA integers. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, volume 10346 of *Lecture Notes in Computer Science*, pages 347–363. Springer, 2017. 3, 6

[EHK04] Mark Ettinger, Peter Høyer, and Emanuel Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Inf. Process. Lett.*, 91(1):43–48, 2004. 7

[Eke19] Martin Ekerå. Revisiting Shor's quantum algorithm for computing general discrete logarithms. *arXiv preprint arXiv:1905.09084*, 2019. 3, 6

[Eke21] Martin Ekerå. Quantum algorithms for computing general discrete logarithms and orders with tradeoffs. *J. Math. Cryptol.*, 15(1):359–407, 2021. 3, 6

[ES21] Edward Eaton and Douglas Stebila. The "quantum annoying" property of password-authenticated key exchange protocols. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings*, volume 12841 of *Lecture Notes in Computer Science*, pages 154–173. Springer, 2021. 5

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985. 3

[GE21] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021. 3

[Gid19] Craig Gidney. Windowed quantum arithmetic. *arXiv preprint arXiv:1905.07682*, 2019. 3

[HJN$^+$20] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. Improved quantum circuits for elliptic curve discrete logarithms. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 425–444. Springer, 2020. 3, 6

[HLS22] Yassine Hamoudi, Qipeng Liu, and Makrand Sinha. Quantum-classical tradeoffs in the random oracle model. *arXiv preprint arXiv:2211.12954*, 2022. 8

[HMR$^+$10] Sean Hallgren, Cristopher Moore, Martin Rötteler, Alexander Russell, and Pranab Sen. Limitations of quantum coset states for graph isomorphism. *J. ACM*, 57(6):34:1–34:33, 2010. 7

[HRT00] Sean Hallgren, Alexander Russell, and Amnon Ta-Shma. Normal subgroup reconstruction and quantum computation using group representations. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 627–635. ACM, 2000. 7

[HS05] Peter Høyer and Robert Spalek. Quantum fan-out is powerful. *Theory Comput.*, 1(1):81–103, 2005. 6

[Joz06] Richard Jozsa. An introduction to measurement based quantum computation. *NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing-From Theory to Experiment*, 199:137–158, 2006. 8

[JQSY19] Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun. General linear group action on tensors: A candidate for post-quantum cryptography. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 251–281. Springer, 2019. 8

[JS08] Tibor Jager and Jörg Schwenk. On the equivalence of generic group models. In *Provable Security: Second International Conference, ProvSec 2008, Shanghai, China, October 30-November 1, 2008. Proceedings 2*, pages 200–209. Springer, 2008. 7

[Kal17] Burton S. Kaliski Jr. A quantum "magic box" for the discrete logarithm problem. Cryptology ePrint Archive, Paper 2017/745, 2017. `http://eprint.iacr.org/2017/745`. 3, 6

[Kit96] Alexei Y. Kitaev. Quantum measurements and the abelian stabilizer problem. *Electron. Colloquium Comput. Complex.*, TR96-003, 1996. 3, 6, 35

[KS01] Fabian Kuhn and René Struik. Random walks revisited: Extensions of Pollard's Rho algorithm for computing multiple discrete logarithms. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography, 8th Annual International Workshop, SAC*

2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers, volume 2259 of *Lecture Notes in Computer Science*, pages 212–229. Springer, 2001. 5

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005. 7

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005. 3, 4, 6, 8, 12, 15

[ME98] Michele Mosca and Artur Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. In Colin P. Williams, editor, *Quantum Computing and Quantum Communications, First NASA International Conference, QCQC'98, Palm Springs, California, USA, February 17-20, 1998, Selected Papers*, volume 1509 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 1998. 3, 6, 35

[MPZ20] Ueli Maurer, Christopher Portmann, and Jiamin Zhu. Unifying generic group models. Cryptology ePrint Archive, Paper 2020/996, 2020. `https://eprint.iacr.org/2020/996`. 7

[MRS08] Cristopher Moore, Alexander Russell, and Leonard J. Schulman. The symmetric group defies strong fourier sampling. *SIAM J. Comput.*, 37(6):1842–1864, 2008. 7

[MZ23] Hart Montgomery and Mark Zhandry. Full quantum equivalence of group action DLog and CDH, and more. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part I*, pages 3–32. Springer, 2023. 8

[Pei20] Chris Peikert. He gives c-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 463–492. Springer, 2020. 7

[PG14] Archimedes Pavlidis and Dimitris Gizopoulos. Fast quantum modular exponentiation architecture for Shor's factoring algorithm. *Quantum Inf. Comput.*, 14(7-8):649–682, 2014. 3

[Pip80] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal on Computing*, 9(2):230–250, 1980. 5, 36

[PZ03] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Inf. Comput.*, 3(4):317–344, 2003. 6

[Reg04] Oded Regev. Quantum computation and lattice problems. *SIAM J. Comput.*, 33(3):738–760, 2004. 7

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin E. Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 241–270. Springer, 2017. 3, 6

[Ros22] Ansis Rosmanis. Hybrid quantum-classical search algorithms. *arXiv preprint arXiv:2202.11443*, 2022. 8

[RS06] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Paper 2006/145, 2006. https://eprint.iacr.org/2006/145. 8

[RS14] Martin Rötteler and Rainer Steinwandt. A quantum circuit to find discrete logarithms on ordinary binary elliptic curves in depth $O(\log^2 n)$. *Quantum Inf. Comput.*, 14(9-10):888–900, 2014. 6

[SBKH93] Kai-Yeung Siu, Jehoshua Bruck, Thomas Kailath, and Thomas Hofmeister. Depth efficient neural networks for division and related problems. *IEEE Trans. Inf. Theory*, 39(3):946–956, 1993. 6

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994. 3, 34

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 256–266. Springer, 1997. 3, 4, 6, 15

[Tho19] Steve Thomas. Re: [Cfrg] proposed PAKE selection process. CFRG Mailing list, June 2019. 5

[Yun15] Aaram Yun. Generic hardness of the multiple discrete logarithm problem. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 817–836. Springer, 2015. 5, 15

[Zha21] Mark Zhandry. Redeeming reset indifferentiability and applications to post-quantum security. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 518–548. Springer, 2021. 7

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 66–96. Springer, 2022. 7, 8, 12