A New Approach to Generic Lower Bounds:
Classical/Quantum MDL, Quantum Factoring, and More

Minki Hhan*

February 17, 2024

Abstract

This paper studies the limitations of the generic approaches to solving cryptographic problems in classical and quantum settings in various models.

- In the classical generic group model (GGM), we find simple alternative proofs for the lower bounds of variants of the discrete logarithm (DL) problem: the multiple-instance DL and one-more DL problems (and their mixture). We also re-prove the unknown-order GGM lower bounds, such as the order finding, root extraction, and repeated squaring.
- In the quantum generic group model (QGGM), we study the complexity of variants of the discrete logarithm. We prove the logarithm DL lower bound in the QGGM even for the composite order setting. We also prove an asymptotically tight lower bound for the multiple-instance DL problem. Both results resolve the open problems suggested in a recent work by Hhan, Yamakawa, and Yun.
- In the quantum generic ring model we newly suggested, we give the logarithmic lower bound for the order-finding algorithms, an important step for Shor's algorithm. We also give a logarithmic lower bound for a certain generic factoring algorithm outputting relatively small integers, which includes a modified version of Regev's algorithm.
- Finally, we prove a lower bound for the basic index calculus method for solving the DL problem in a new idealized group model regarding smooth numbers.

The quantum lower bounds in both models allow certain (different) types of classical preprocessing.

All of the proofs are significantly simpler than the previous proofs and are through a single tool, the so-called compression lemma, along with linear algebra tools. Our use of this lemma may be of independent interest.

*E-mail:minkihhan@gmail.com. KIAS

Contents

1 Introduction

What is the source of the generic hardness of some cryptographic problems?

The generic group models (GGM) [Nec94, Sho97, Mau05] are the most successful and influential idealized models in cryptography. In this model, the group operations can be carried out by making queries to a group oracle, and any other use of the particular features of the group is not allowed. Despite its restricted nature, many important algorithms, such as Pohlig-Hellman [PH78] or Pollard's rho algorithm [Pol78], are encompassed by the class of generic group algorithms. Despite some criticisms [Den02, KM06] and non-generic algorithms, e.g., index-calculus, the GGM plays an important test bed for cryptographic protocols, and the security proofs in the GGM provide a sanity check guaranteeing that there are no simple attacks. The proofs in the GGM become more meaningful in the elliptic-curve groups.

The GGM is especially promising because of its simple security proofs; most of the security proofs in the GGM heavily rely on the Schwartz-Zippel (SZ) lemma that is already used in [Sho97]. This lemma roughly states that for a non-zero multivariate linear polynomial P over $\mathbb{Z}_p$ for a prime p, the probability that a random element becomes a root of P is $1/p$. This provides a meaningful limitation for generic algorithms obtaining a single piece of information and is used to prove the generic lower bounds for the discrete logarithm (DL) problem and computational/decisional Diffie-Hellman (C/DDH) problems, as well as many cryptographic applications.

We face hurdles in proving the generic security when we slightly tweak the model or problems. If we consider the unknown-order groups, the lower bounds for various problems can be proven with relatively small efforts, including the order-finding problem [Sut07] or the root extraction problems [DK02]. When we consider the problems where enormous amounts of information can be obtained, the security proofs based on the SZ lemma do not work. To remedy this, other idealized problems (e.g., the search-by-hyperplane/surfaces) [Yun15, AGK20, AHP23] are suggested or seemingly involved techniques (e.g., compression lemmas or pre-sampling) are used [CK18, CDG18] in the proofs. In the quantum setting, the rigorous proofs are rather complicated and pass through the classical lower bounds [HYY23].

Extensions beyond the group structures [BL96, BV98, AM09, JS13, YYHK20] become much more complicated. In many cases, there is some evidence that the unconditional lower bounds unlikely exist, and the proofs are done through the reduction between the problems. To our knowledge, there are no known unconditional lower bounds, even in the idealized models.

This state of affairs makes the genuine source of the generic hardness elusive and asks for case-by-case studies for each model. In particular, the unconditional lower bounds in idealized settings are only known for the generic group models.

1.1 Our Results

We provide a unified way to prove the old and new hardness proofs in the various idealized models: the known/unknown-order classical generic groups, quantum generic groups, quantum generic *rings*, and the new group model embracing index calculus.

Our main technical lemma is, along with some linear algebraic observations, (variants of) *the compression lemma*, which roughly asserts that there is no way to compress n-bit strings to strings less than n-bit. This lemma is occasionally used in proving the time-space tradeoff lower bounds [DTT10, NABT15, DGK17, CK18, HXY19, CLQ19], and introduces a highly involved proof. Our proofs are significantly simpler, as shown in this section. Roughly, we compress the problem instances along with some relevant randomness into the information that generic algorithms can obtain; the decoding simulates the generic algorithm using the encoding, *without* accessing the oracles, but still recovers the problem instances. This gives some clues that the generic hardness is from the limited way of obtaining information on generic algorithms.

This paper mainly focuses on the abstract model of Maurer [Mau05]. In this model, the group (or ring) elements are stored in *element* wires, and they can be accessed only by the group operation gates or the equality gates.

The Known-order GGM Lower Bounds Let us begin with the lower bound of folklore for the DL problem (Theorem 3.3). Let $\mathcal{G} \simeq \mathbb{Z}_p$ be the underlying group of prime order p. In this problem, the algorithm $\mathcal{A}$ is given (g, g^x) and is asked to find x. Suppose that $\mathcal{A}$ solves the DL problem with almost certainty with T group operations. We also associate a polynomial $aX + b$ to the group element g^{ax+b}. It is not hard to argue that (slightly modified) $\mathcal{A}$ finds two equal group elements with different polynomials.

We use this algorithm to compress $x \in [p]$. Among T group elements, there are T^2 possibilities for a pair of equal group elements. In other words, we can encode the discrete logarithm x in T^2 possible collisions, and the compression lemma says that $\log T^2 = 2 \log T \geq \log p$ for the high success probability. This implies that $T \geq \sqrt{|\mathcal{G}|}$ as in the previous proofs.

We proceed to the MDL problem (Theorem 3.4). Suppose that the algorithm $\mathcal{A}$ with T group operations is given $(g, g^{x_1}, ..., g^{x_m})$ and is asked to find $\mathbf{x} = (x_1, ..., x_m)$. As before, we can assume that $\mathcal{A}$ finds m collisions. We can encode $\mathbf{x}$ using the information of the collisions, which requires

$$\log \binom{\binom{T}{2}}{m} \approx m \log \frac{eT^2}{2m}$$

bits. To solve the MDL problem with certainty, it must be larger than $m \log |\mathcal{G}|$, which is the information that $\mathbf{x}$ possesses, implying that $T \geq \sqrt{mp}$. Previously, this bound was first proven in [Yun15] and required an involved argument regarding the related problem called the search-by-hyperplane-queries (SHQ). We note that we have *another* simple proof for this lower bound solely based on the linear-algebra reasoning in Theorem B.1.

The same proof strategy easily extends to the other problems. This includes the gap-DL and gap-CDH problems (Theorems 3.5 and 3.6) and the one-more DL problem (OM-DL) (Theorem 3.7) that was first proven recently [BFP21] (and was falsely proven in [CDG18]). We actually prove the lower bounds for a much more general problem, where the adversary is asked to find the n-more DL solutions than its queries to the DL oracles; $n = 1$ corresponds to the OM-DL problem.

The Unknown-order GGM Lower Bounds We also consider the unknown-order GGM. In Section 4, we show that the same strategy can prove the lower bounds for the order-finding in the prime-order group (Theorem 4.1) that is shown in [Sut07] and in the RSA group (Theorem 4.2). We also prove the hardness of the root extraction (Theorem 4.3), which was proven in [DK02], and the repeated squaring (Theorem 4.4) in the unknown-order GGM. We stress that we do *not* consider the ring operations. Thus, its implications are limited to the group setting.

We sketch the proof for the order-finding problem. In this model, the generic algorithm can compute $g^{x \pm y}$ for given g^x, g^y as in the previous GGM, but does not know the order of the underlying group. Therefore, the corresponding polynomials have a bounded coefficient after T group operations, so the number of their prime factors is bounded. It turns out that each equality gate can contain T prime divisors. The encoding contains the equality gate that specifies the order, and the index of its divisors. The length becomes $3 \log T$ to compress $\log |\mathcal{G}|$-bit order, giving the $T \geq |\mathcal{G}|^{1/3}$ bound.

The Quantum GGM Lower Bounds We prove the quantum lower bounds for solving the DL problem (Theorem 5.2) and variants in the quantum GGM (QGGM). This direction was suggested in [HYY23], and the authors gave the lower bounds for the DL and C/DDH problems in the QGGM.

The proof strategy is different from the classical lower bounds. Instead of the one-shot encoding as in the classical setting, we need an interactive version of the compression lemma (Corollary 2.3) proven in [HNR18]. This roughly states that if Alice wants to send an n-bit message to Bob, Alice needs to send n-bit anyway, regardless of the amounts of Bob's messages to Alice and the number of rounds.

In the QGGM, the algorithm can make group operations coherently. Given a generic algorithm, we construct the interactive protocol between Alice and Bob, where Alice holds all the group elements, and Bob holds the other registers. Bob runs the DL algorithm, and whenever it needs to make a quantum group operation, he sends the relevant registers to Alice; Alice applies the group operations and returns

the relevant registers to Bob. For simplicity, we assume that the indices for the target group elements are classical. In this setting, Alice sends two bits (or one qubit) to delegate the group operation Bob requested. If the algorithm makes Q group operations, the interactive version of the compression lemmas proves $2Q \geq \log |\mathcal{G}|$, recovering the previous lower bound; actually, with a better constant than the previous bound $4Q \geq \log |\mathcal{G}|$.

If we allow the indices to be quantum, delegating quantum group operations requires more communication to include them. The lower bound becomes $Q = \Omega(\log |\mathcal{G}|/ \log \ell)$ for the length of indices ℓ. The same strategy naturally extends to the MDL problem (Theorem 5.3) in the QGGM, proving $Q = \Omega(m \log G/ \log \ell.)$

Our proof equally works for the *composite order* DL problems and holds even regarding the classical preprocessing. The composite order DL lower bound and MDL lower bound in the QGGM resolves the open problems asked in [HYY23], where the matching algorithms were suggested. In fact, our lower bound implies that the number of quantumly accessible indices is an important measure, while the previous results only consider the memory-bounded setting, which naturally bounds the number of quantum indices. Also, our lower bound implies that the speed-up for the MDL problem beyond Shor in terms of the group operation complexity requires a large quantum data structure.

We also prove the QGGM variant for the order-finding problems (Theorem 5.4), showing the order-finding in the QGGM requires $\Omega(\log |\mathcal{G}|)$ quantum group operations, even with classical preprocessing.

The Quantum Generic Ring Model and Lower Bounds We study a quantum variant of the generic ring model [AM09, JS13], which we call the quantum generic ring model (QGRM). In this model, the algorithm has oracle access to the ring elements as in the GGM. However, we do *not* give the explicit value of N to the algorithm because we aim for the unconditional lower bounds in the idealized model. If the algorithm knows N, we cannot rule out the direct use of N, and the proof must be through reductions as in [AM09].

We prove that the logarithmic lower bound for the QGRM order finding algorithm in the ring isomorphic to $\mathbb{Z}_N$ where N is a product of two safe primes (Theorem 6.1). The order (or period) finding problem is a major subroutine in Shor's factoring algorithm [Sho99].

Note that a recent work of Regev [Reg23] solves the integer factorization with a different method. In this approach, small integers are extensively used, taking advantage of the fact that small integer arithmetic operations are faster than large integer operations, giving an improved algorithm with better circuit complexity.

We observe that this advantage results in a modified algorithm that outputs a plain integer with a non-trivial common factor with N of relatively small size. We consider the generic algorithms that output such an integer to solve the integer factoring that can be computed *without* modulus reductions—this must be done in plain because QGRM algorithms do not know N. We prove that if the output is relatively small, the logarithmic ring operation lower bound holds for factoring (Theorem 6.2). Intriguingly, the output of Shor's algorithm with this modification is too large to apply this lower bound.

These results give the first evidence that the quantum factoring algorithm needs a logarithmic number of group operations. Our result extends to the straight-line classical preprocessing that reflects the real world better.

Beyond GGMs: Index Calculus Finally, we study the idealized group model, called the smooth GGM, beyond the generic groups, encompassing the index calculus method. This model provides the abstraction for the notion of smooth elements and efficient factoring for the smooth integers.

We prove that the DL algorithm must make $\exp\left(C\sqrt{\log |\mathcal{G}| \log\log |\mathcal{G}|}\right)$ group operations for some constant $C > 0$ in the SGGM (Theorem 7.1), giving some evidence that going beyond this bound requires a new idea, as the ones in the number field sieves.

We do not claim this lower bound provides new insights or strong evidence for the index calculus. We believe that the ideas used in the proof for the SGGM lower bound must have been observed and used in the development of the index calculus, especially for optimization. Still, our result shows that a proper

abstraction of the generic approaches, where only limited operations are used, can indeed prove that these approaches cannot go further; asking for new ideas.

Notations. For a positive integer N, a finite cyclic group of order N is denoted by $\mathbb{Z}_N$, identified by $\{0, 1, ..., N-1\}$ with the natural group operation, and $[N] := \{1, ..., N\}$.

2 Compression Lemmas

This section presents our main lemmas, which are usually called the compression lemma. The classical compression lemma is stated as follows.

Lemma 2.1. *Let $\mathcal{M}, R$ be finite sets. Let* $\mathsf{Encode} : \mathcal{M} \times R \to \{0,1\}^m$ *and* $\mathsf{Decode} : \{0,1\}^m \times R \to \mathcal{M}$ *be deterministic algorithms. For $\epsilon \in (0, 1]$, if*

$$\Pr_{r \leftarrow R, x \leftarrow \mathcal{M}}[\mathsf{Decode}(\mathsf{Encode}(x, r), r) = x] \geq \epsilon,$$

then we have $m \geq \log |\mathcal{M}| + \log \epsilon$.

This is a direct corollary of the following quantum interactive version of the compression lemma. Precisely, the classical one-way protocol with the preshared entanglement $\sum_{r \in R} |r, r\rangle$ corresponds to the above lemma.

Lemma 2.2 ([HNR18, Theorem 1.2])**.** *Consider an interactive protocol between Alice and Bob, who share an arbitrarily entangled state and communicate through classical channels. Alice wants to send a uniformly random element in a finite set $\mathcal{M}$ to Bob. Suppose that the probability that Bob correctly recovers x with probability $\epsilon \in (0, 1]$, and Alice sends m bits to Bob total over all rounds. Then it holds that $m \geq \log |\mathcal{M}| + \log \epsilon$, regardless of the number of bits sent by Bob to Alice.*

In general, if Alice sends a classical string in $[M_i]$ to Bob as the i-th round message for $i \in [k]$ where k is the maximum number of rounds, then it holds that

$$\log \left(\prod_{i=1}^{k} M_i \right) \geq \log |\mathcal{M}| + \log \epsilon.$$

The original theorem in [HNR18, Theorem 1.2] mainly concerns the case of $\mathcal{M} = \{0,1\}^n$ and the bit-strings as messages. This generalization is straightforward.[1] The quantum communication version can be derived using quantum teleportation (See also [NS06, Theorem 2]).

Corollary 2.3. *In the same setting as the above lemma, if Alice and Bob can communicate through* quantum channels, *the bounds become*

$$m \geq \frac{\log |\mathcal{M}| + \log \epsilon}{2}, \quad \textit{and} \quad \log \left(\prod_{i=1}^{k} M_i \right) \geq \frac{\log |\mathcal{M}| + \log \epsilon}{2},$$

respectively, where Alice sends one qudit of dimension M_i in the i-th round. When Alice additionally sends c classical bits, the bounds become

$$2m + c \geq \log |\mathcal{M}| + \log \epsilon, \quad \textit{and} \quad 2 \log \left(\prod_{i=1}^{k} M_i \right) + c \geq \log |\mathcal{M}| + \log \epsilon.$$

We give some remarks. The above lemmas consider the average-case probability for input x, while the previous (both classical and quantum) versions [GT00, DTT10, NS06] consider the case that the success probability is at least ϵ for any input x. This caused a significant loss in the resulting security in the first AI-QROM bound [HXY19], or call for the random-self-reducibility in the preprocessing DL security [CK18]. Thanks to this average-case feature, we exclude the random-self-reducibility in the proofs.

[1] Roughly, the choice of $\mathcal{M} = \{0,1\}^n$ is only used at the end of the proof where the probability that input to Alice is x is $1/2^n$, and modifying it to $1/|\mathcal{M}|$ suffices to prove our theorem. The non-bit-string is slightly involved, but changing the appropriate set suffices.

3 Lower Bounds in the Classical Generic Group Model

3.1 Generic Group Model

We first define the generic group model (GGM) of Maurer [Mau05], also known as the type-safe model [Zha22]. Let N be the known prime order[2] of our interested finite cyclic group $\mathcal{G} \cong \mathbb{Z}_N$ with a generator g. A generic algorithm $\mathcal{A}$ in this model is given by a circuit with the following features:

- There are two types of wires: bit wires and (group) element wires. Bit wires take a bit in $\{0,1\}$, whereas element wires take an element in $\mathbb{Z}_N \cup \{\perp\}$. For an element wire containing x, we write g^x to denote this wire to distinguish it from a bit string.
- There are bit gates that map bits to bits, which cannot take element wires as input.
- There are three special gates called *element gates* that can access the element wires as follows:

  Labeling Gate. It takes $\lceil \log_2 N \rceil$ bit wires and interprets them as an element in $x \in \mathbb{Z}_N$ as input, and outputs an element wire g^x. If there is no corresponding element $x \in \mathbb{Z}_N$ to the input wires, it outputs an element wire containing $\perp$.

  Group Operation Gate. It takes two element wires containing g^x, g^y and a single bit wire containing b as input. If both g^x, g^y are not $\perp$, it outputs an element wire containing g^{x+by}.[3] Otherwise, it outputs an element wire containing $\perp$.

  Equality Gate. It takes two element wires as input. If both wires contain the same element $g^x \neq \perp$, it outputs a bit wire containing 1. In all other cases including $\perp$ inputs, the output is 0.

An algorithm $\mathcal{A}$ in this model is called a GGM algorithm and is usually denoted by $\mathcal{A}^{\mathcal{G}}$. The cost metric for the algorithms, denoted by *the group operation complexity*, counts the number of labeling and group operation gates used in the circuit, and all other gates are considered free.

We assume that the element gates have some orders so they can be applied sequentially (along with required bit gates).[4] We also assume that GGM algorithms never make two equality gates with the same input wires. This ensures that for a GGM algorithm taking m element wires as input and with the group operation complexity T, the number of group operation gates T, the number of equality gates less than or equal to $\binom{m+T}{2}$. We further assume that the description of the GGM algorithm contains the order of the element gates so that they can be applied in order (ignoring bit gates).

Remark 1 (Relations to the other generic group models.)**.** A different model for generic group algorithms is suggested by Shoup [Sho97]. The results for known-order GGM algorithms in this paper can be extended to Shoup's generic group model. This is because this paper focuses on the cryptographic assumptions that can be described as a single-stage game, where the generic equivalence between two models is known [Zha22]. We place the detailed theorem with the proof in Appendix C for completeness. We note that our proof can be extended to the Shoup-style GGM directly, as shown in Appendix C.1. However, this makes the proof involved, and the main body focuses on the Maurer-style model for a simpler exposition.

3.1.1 Variations: Maintaining Polynomials

Before proceeding to the classical lower bounds in the generic group model, we give a variation of GGM algorithms, which maintains the polynomials representing the elements and information that it achieved. We assume that the input to the GGM algorithm is specified by polynomials $P_1, ..., P_m \in \mathbb{Z}_N[X_1, ..., X_t]$ for some formal variables $X_1, ..., X_t$ corresponding to the hidden values. For example, in the discrete logarithm problem, X_1 specifies the problem instance g^x, and the input is specified by $P_1 = 1, P_2 = X_1$. We

[2] We can extend to the composite-order setting easily.

[3] One may define this gate differently, e.g., $(g^x, g^y, a, b) \mapsto g^{ax+by}$, but it does not make any change to our result.

[4] Given the circuit, such an order can be found using the breadth-first search.

occasionally identify a polynomial $P = a_1X_1 + ... + a_tX_t + b$ as a vector $(b, a_1, ..., a_t) \in \mathbb{Z}_N^{t+1}$ (recall N is prime, which makes $\mathbb{Z}_N^{t+1}$ a vector space.) especially when we discuss the linear algebra notions.

Given the polynomial representations of inputs, we maintain a list $\mathcal{P}$ of a pair of the element wire and polynomial called *the polynomial list* and a counter c, and it behaves as follows.

- As an initialization, set $\mathcal{P}$ as an empty list. For each input element wire w containing a group element corresponding P_i for $i \in [m]$, store (w, P_i) in the i-th row of $\mathcal{P}$. Set $c \leftarrow m$.
- For a labeling gate in the circuit of $\mathcal{A}$ with input representing $a \in \mathbb{Z}_N$ and output element wire w, set $c \leftarrow c + 1$, and store $(w, P_c := a)$ in the c-th row of $\mathcal{P}$.
- For a group operation gate with two element wires w_1, w_2 and a bit wire containing b as input and output wire w appears, find $i, j \leq c$ such that i, j-th rows of $\mathcal{P}$ are w_1, w_2. Set $c \leftarrow c + 1$, compute $P_c := P_i + (-1)^b P_j$, and store (w, P_c) in the c-th row of $\mathcal{P}$.

The equality gates are dealt with differently, by maintaining *the zero sets* $\mathcal{Z}$ that is initialized as an empty set. For an equality gate eq with two input element wires w_1, w_2 and output 1 (i.e., they are equal), we find i, j-th rows of $\mathcal{P}$ containing w_1, w_2 and call g by *collision*; since no two equality gates have the same inputs, we also call (i, j) as a collision ambiguously. We process each collision as follows. We do nothing for the equality gates outputting 0.

- If $P_i = P_j$ as a polynomial over $\mathbb{Z}_N$, then the collision is called *trivial*, and do nothing.
- If an equality query finds a nontrivial collision (i, j), then write $|\mathcal{Z}| = z$ and $\mathcal{Z} = \{Q_i\}_{i \in [z]}$, check if there exists $\mathbf{a} = (a_1, ..., a_z) \in \mathbb{Z}_N^z$ such that

$$P_i - P_j = a_1Q_1 + ... + a_zQ_z \tag{1}$$

  holds as a polynomial. If there is no such $\mathbf{a}$, updates $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{P_i - P_j\}$. We call the collision (i, j) *informative*, and otherwise *predictable.*

Note that the notion of informative collisions is similar to the *useful* queries in [Yun15] in the search-by-hyperplane problem, but our definition is purely linear-algebraic and direct. It just says that the new informative collision must not be included in the span of the previous collisions.

The informative collisions are sufficient for describing the behavior of the GGM algorithm, as shown in the following lemma, proved in Appendix A.1.

Lemma 3.1. *Let $\mathcal{A}$ be a GGM algorithm. Given a description of the circuit for $\mathcal{A}$ and the zero set $\mathcal{Z}$ for the given input, the polynomial list of $\mathcal{A}$ right before its termination can be computed without using the element gates, i.e., computed by a Boolean circuit.*

The following auxiliary lemma is a generalization of the Schwartz-Zippel lemma, which could be of independent interest. It gives another alternative proof for the MDL lower bound presented in Appendix B with its proof.

Lemma 3.2. *Suppose the hidden variables $x_1, ..., x_t$ are uniform in $\mathbb{Z}_N$, and the group elements during the execution of the algorithm always correspond to the linear polynomials in $\mathbb{Z}_N[X_1, ..., X_t]$. For any equality gate for w_i, w_j, the probability that it induces an informative collision is at most $1/N$.*

3.2 The Discrete Logarithm Problem and Friends

We first prove the following well-known generic lower bound for the DL problem.

Problem 1. A discrete logarithm (DL) problem for a cyclic group $\mathcal{G}$ of order p with a generator g asks to find x given (g, g^x) for uniformly random $x \in \{0, ..., p-1\}$. An m-multiple DL (m-MDL) problem asks to find $\mathbf{x} = (x_1, ..., x_m)$ given $(g, g^{x_1}, ..., g^{x_m})$ for uniformly random $\mathbf{x} \in \{0, ..., p-1\}^m$. In the (Q)GGM, the group is fixed a priori, and the inputs are stored in the element registers.

Theorem 3.3. *Let $\mathcal{G}$ be a cyclic group of prime order. Let $\mathcal{A}_{\mathsf{DL}}$ be a DL algorithm in the GGM having at most T group operation gates, then the following holds:*

$$\Pr_{\mathcal{A}_{\mathsf{DL}},x}\left[\mathcal{A}^{\mathcal{G}}_{\mathsf{DL}}(g,g^x)\to x\right]=O\left(\frac{T^2}{|\mathcal{G}|}\right).$$

Proof. Let $p=|\mathcal{G}|$ and ϵ be the success probability of $\mathcal{A}_{\mathsf{DL}}$. We make the following modifications: For $z\leftarrow\mathcal{A}^{\mathcal{G}}_{\mathsf{DL}}(g,g^x)$, we let the algorithm make the labeling gate on input z and apply the equality gate on input (g^z,g^x) to find a collision at the end, so that $\mathcal{A}$ always finds an informative collision with probability at least ϵ. Including this procedure, we assume that the algorithm makes $C=T+1$ group operations. The algorithm $\mathcal{A}_{\mathsf{DL}}$ may be randomized by taking a random string r as a seed.

Now, we construct a pair of encoding and decoding protocols for $\mathcal{M}=[p]$ and a set R of seed r. For $x\in[p]$, the protocols are defined as follows.

$\mathsf{Encode}(x,r)$**:** It runs $\mathcal{A}^{\mathcal{G}}_{\mathsf{DL}}(g,g^x)$ with randomness r and outputs the equality gate c with input (i,j) that is the lexicographically first informative collision, i.e., for any other informative collision (i',j'), it holds that $i<i'$, or $i=i'$ and $j<j'$. If there is no informative collision, it outputs a special symbol $c=\perp$.

$\mathsf{Decode}(c,r)$**:** If $c=\perp$, it outputs a random value in $[p]$. Otherwise, it constructs a sub-circuit $\mathcal{A}'_{\mathsf{DL}}$ of $\mathcal{A}_{\mathsf{DL}}$ by cutting out the gates after the equality gate c corresponding to the first informative collision (i,j). We associate the group element g^{ax+b} with a polynomial $aX+b\in\mathbb{Z}_p[X]$. By Lemma 3.1, the corresponding polynomials $P_i=a_iX+b_i$ and $P_j=a_jX+b_j$ can be computed without using the element wires. Then it returns $z=-(b_i-b_j)/(a_i-a_j) \bmod p$ as an output.

We prove this protocol is correct with a probability of at least ϵ, or whenever $\mathcal{A}_{\mathsf{DL}}$ finds x. In this case, the encoder finds a collision c with input (i,j), which is informative only when

$$(a_ix+b_i=a_jx+b_j \bmod p)\wedge((a_i,b_i)\neq(a_j,b_j)) \iff x=-\frac{b_i-b_j}{a_i-a_j} \bmod p.$$

Thus, given $c\neq\perp$, the decoder always finds the correct answer x, i.e., the protocol succeeds with probability at least ϵ.

Now we compute the encoding length of the protocol. Since $\mathcal{A}_{\mathsf{DL}}$ obtains at most $C+2$ group elements including inputs, the encoding space $\mathcal{C}$ has the cardinality $\binom{C+2}{2}+1\le(T+3)^2/2$. By Lemma 2.1, we have the following inequality

$$\log\epsilon+\log|\mathcal{G}|\le\log|\mathcal{C}|\le\log\left(\frac{(T+3)^2}{2}\right)\implies\epsilon=O\left(\frac{T^2}{|\mathcal{G}|}\right)$$

which concludes the proof. □

It can easily be extended to the multiple-instance DL problem with small adjustments. For a positive integer m, we write $g^{\mathbf{x}}$ to denote $(g^{x_1},...,g^{x_m})$.

Theorem 3.4. *Let $\mathcal{G}$ be a cyclic group of prime order. Let $\mathcal{A}_{m\text{-}\mathsf{MDL}}$ be an m-MDL algorithm in the GGM having at most T group operation gates. It holds that:*

$$\Pr_{\mathcal{A}_{m\text{-}\mathsf{MDL}},\mathbf{x}}\left[\mathcal{A}^{\mathcal{G}}_{m\text{-}\mathsf{MDL}}(g,g^{\mathbf{x}})\to\mathbf{x}\right]=O\left(\left(\frac{e(T+2m+1)^2}{2m|\mathcal{G}|}\right)^m\right).$$

Proof. Let $p=|\mathcal{G}|$ and ϵ be the success probability of $\mathcal{A}_{m\text{-}\mathsf{MDL}}$. With a similar modification, we assume that $\mathcal{A}_{m\text{-}\mathsf{MDL}}$ finds at least m informative collisions with probability at least ϵ using $C=T+m$ group operation complexity. We associate a group element $g^{a_1x_1+...+a_mx_m+b}$ with a polynomial $a_1X_1+...+a_mX_m+b\in\mathbb{Z}_p[X_1,...,X_m]$. We additionally need the following result from linear algebra.

Fact 1. Given m informative collisions, there is a polynomial time algorithm to find the unique assignments $(X_1, ..., X_m) = (x_1, ..., x_m)$ making the given collisions informative.

The proof can be found in Appendix A.1.

$\mathcal{A}_{m\text{-MDL}}$ may be randomized using a random seed r. We construct encoding and decoding protocols for $\mathcal{M} = [p]^m$ and a set R of the seed r. For input $\mathbf{x} \in \mathbb{Z}_p^m$, the protocols are defined as follows.

Encode$(\mathbf{x}, r)$**:** It runs $\mathcal{A}^{\mathcal{G}}_{m\text{-MDL}}(g, g^{\mathbf{x}})$ with randomness r and collects the lexicographically first m informative collision gates c_k with input (i_k, j_k) for $k \in [m]$. If m informative collisions are found during the execution, it outputs $\mathbf{c} = (c_1, ..., c_m)$. Otherwise, it outputs a symbol $\perp$.

Decode$(\mathbf{c}, r)$**:** If $\mathbf{c} = \perp$, it outputs a random value in $[p]^m$. Otherwise, it parses $\mathbf{c} = (c_1, ..., c_m)$ and constructs a sub-circuit $\mathcal{A}'_{m\text{-MDL}}$ of $\mathcal{A}_{m\text{-MDL}}$ by cutting out the gates after the m-th informative collision gate (corresponding to c_m). By Lemma 3.1, it recovers the polynomial list. Using $\mathbf{c}$ and Fact 1, it finds and outputs the assignment $\mathbf{z} = (z_1, ..., z_m)$ of $(X_1, ..., X_m)$.

It is obvious that if $\mathcal{A}_{m\text{-MDL}}$ finds $\mathbf{x} = (x_1, ..., x_m)$ then the decoder correctly recovers $\mathbf{x}$, which happens with probability at least ϵ. We focus on the encoding size below. Let $B = \binom{C+m+1}{2}$ be the upper bound of the number of equality queries. The bit-length for describing the m informative collision (or $\perp$) is less than $\log\left(\binom{B}{m} + 1\right)$, which is bounded by

$$\log \binom{B+1}{m} \leq m \log\left(\frac{e(B+1)}{m}\right) \leq m \log\left(\frac{e(T+2m+1)^2}{2m}\right),$$

where we use $B + 1 \leq \frac{(C+m+1)^2}{2} \leq \frac{(T+2m+1)^2}{2}$. By Lemma 2.1, we have

$$\log \epsilon + m \log |\mathcal{G}| \leq \log |\mathcal{C}| \leq m \log\left(\frac{e(T+2m+1)^2}{2m}\right)$$

which can be rewritten as follows

$$\epsilon = O\left(\left(\frac{e(T+2m+1)^2}{2m|\mathcal{G}|}\right)^m\right),$$

as we desired. □

3.3 Oracle Problems in the GGM

This section extends the lower bounds relative to the oracle. We first consider the following problems.

Problem 2. In the gap DL (gap-DL) problem, the adversary is given (g, g^x) as input and is asked to find x, having access to the decisional Diffie-Hellman (DDH) oracle: $O_{\mathsf{DDH}} : (g^x, g^y, g^z) \mapsto \delta_{xy,z}$. In the gap computational Diffie-Hellman (gap-CDH) problem, the adversary is given (g, g^x, g^y) and is asked to output g^{xy} with the DDH oracle access.

Theorem 3.5. *Let $\mathcal{G}$ be a cyclic group. Let $\mathcal{A}_{\mathsf{Gap\text{-}DL}}$ be a gap-DL algorithm in the GGM having at most T group operation gates and making T_{DDH} queries to the DDH oracle, then the following holds:*

$$\Pr_{\mathcal{A}_{\mathsf{Gap\text{-}DL}}, x}\left[\mathcal{A}^{\mathcal{G}, O_{\mathsf{DDH}}}_{\mathsf{Gap\text{-}DL}}(g, g^x) \to x\right] = O\left(\frac{T^2 + T_{\mathsf{DDH}}}{|\mathcal{G}|}\right).$$

Proof sketch. We extend the notion of collisions to include the DDH oracle answers that output 1. By a similar modification as the previous section, we can assume that the algorithm finds at least one informative collision. If it is an answer from the DDH oracle, then it specifies the equation $(aX + b)(cX + d) = eX + f$ for some a, b, c, d, e, f. It has at most two solutions; thus, the encoding includes one more bit to specify the correct solution. The number of collisions is bounded by $\binom{T+3}{2} + T_{\mathsf{DDH}}$. The other parts of the proof are identical. □

Theorem 3.6. *Let $\mathcal{G}$ be a cyclic group. Let $\mathcal{A}_{\mathsf{Gap\text{-}CDH}}$ be a gap-CDH algorithm in the GGM having at most T group operation gates and making T_{DDH} queries to the DDH oracle, then the following holds:*

$$\Pr_{\mathcal{A}_{\mathsf{Gap\text{-}CDH}},x}\left[\mathcal{A}^{\mathcal{G},O_{\mathsf{DDH}}}_{\mathsf{Gap\text{-}CDH}}(g,g^x,g^y)\to g^{xy}\right]=O\left(\frac{T^2+T_{\mathsf{DDH}}}{|\mathcal{G}|}\right).$$

The proof is almost identical and placed in Appendix A.1.

Problem 3. In the one-more-DL (OM-DL) problem, the adversary is given access to the challenge oracle O_{Chal} that outputs g^{x_i} for an unknown x_i and to the DL oracle $O_{\mathsf{DL}}: g^x \mapsto x$. The number of DL oracle queries q must be less than the number of challenge queries t. The adversary aims to find all answers to the challenges. More generally, in the n-out-of-m-more-DL ((n,m)-M-DL) problem, it must hold that $t=q+m$, and the adversary needs to find $q+n$ solutions to the challenges among $q+m$ challenges.

Theorem 3.7. *Let $\mathcal{G}$ be a cyclic group. Let $\mathcal{A}_{(m,n)\text{-}\mathsf{M\text{-}DL}}$ be an n-out-of-m-more-DL algorithm in the GGM having at most T group operation gates and making q queries to the DL oracle, then the following holds:*

$$\Pr_{\mathcal{A}_{(n,m)\text{-}\mathsf{M\text{-}DL}},x}\left[\mathcal{A}^{\mathcal{G},O_{\mathsf{Chal}},O_{\mathsf{DL}}}_{(n,m)\text{-}\mathsf{M\text{-}DL}}(g)\ \textit{solves}\ (n,m)\text{-}\mathsf{M\text{-}DL}\right]=O\left(\left(\frac{e(T+m+n+1)^2}{|\mathcal{G}|}\right)^n\right).$$

In particular, the advantage against OM-DL *is* $O\left(\frac{T^2}{|\mathcal{G}|}\right)$.

Proof. Suppose that the $t=m+q$ challenges are $g^{x_1},...,g^{x_t}$. We construct an encoding for $\mathbf{x}=(x_1,...,x_t)$. We assume that the algorithm is deterministic. Regarding informative collisions, we include the DL oracle answers as the collision. If the DL oracle outputs z for input g^P, we regard $P-z$ as a collision. Since the algorithm finds $n+q$ solutions, it must find $n+q$ informative collisions (including the DL oracle outputs). We assume that the algorithm never queries to the DL oracle that the answer induces a trivial collision. This means there are n informative collisions that are not from the DL oracle queries.

We need the following simple fact from linear algebra.

Fact 2. Given a linear independent linear equations over b variables for $a<b$. There are $b-a$ variables such that the linear equations are still independent after fixing them to some values.

The procedures are as follows.

$\mathsf{Encode}(\mathbf{x})$**:** It runs $\mathcal{A}^{\mathcal{G},O_{\mathsf{Chal}},O_{\mathsf{DL}}}_{m\text{-}\mathsf{M\text{-}DL}}(g)$ and collects the lexicographically first t informative collisions, which could be the equality gate or the DL oracle answer. If $n+q$ informative collisions are found during the execution, it outputs $\mathbf{c}=(c_1,...,c_n)$ that denote the informative equality gates and the DL oracle answers $\mathbf{z}=(z_1,...,z_q)$. By Fact 2, there are $m-n$ x_i's such that revealing them does not hurt the linear independence of the informative collisions. Finally, those x_i's, denoted by $\mathbf{w}$ become a part of the encoding. Otherwise, it outputs a symbol $\perp$.

$\mathsf{Decode}(\mathbf{c},\mathbf{z},\mathbf{w})$**:** It runs $\mathcal{A}^{\mathcal{G},O_{\mathsf{Chal}},O_{\mathsf{DL}}}_{m\text{-}\mathsf{M\text{-}DL}}(g)$ to recover $n+q$ informative collisions. Given these equations, the decoder can recognize the indices for $\mathbf{w}$. It recovers $\mathbf{w}$, and plugs them in the informative collisions. The informative collisions become $n+q$ linear equations over $n+q$ variables, so that it can recover $\mathbf{x}$.

The length of the encoding is bounded by

$$\log\binom{\binom{T+m+n}{2}}{n}+q\log|\mathcal{G}|+(m-n)\log|\mathcal{G}|+O(1)$$

which must be larger than $(m+q)\log|\mathcal{G}|+\log\epsilon$ for the success probability ϵ by Lemma 2.1. This gives

$$n\log\left(\frac{e(T+2m+1)^2}{2n}\right)\geq n\log|\mathcal{G}|+\log\epsilon$$

which implies

$$\epsilon = O\left(\left(\frac{e(T+m+n+1)^2}{|\mathcal{G}|}\right)^n\right),$$

concluding the proof. □

Remark 2. Extending the results to high-degree variants like m-CDH problems is not trivial. We believe with some algebraic geometry reasoning like Bézout theorem, as in [AGK20], the high-degree variants can be proven in essentially the same way.

4 Lower Bounds in the Unknown-order GGM

We extend the generic group to the unknown-order setting. As the order is unknown, we should consider the distribution of the order.

Let $\mathcal{G}$ be a cyclic group of order N, where the distribution of N will be specified later. We assume that N is unknown to the algorithm except for its bit length. The other interface of the generic algorithms is identical to the (known-order) GGM. In particular, the assumption that the group operation only allows to compute $(g^x, g^y) \mapsto g^{x+y}$ is important.[5] Note that the algorithm cannot extract any information from the element wire containing $\perp$; for example, the equality gate involving $\perp$ always outputs 0.

Remark 3. We found that the known equivalence proof between the generic group models does not extend to the unknown-order group setting. Therefore, we include the random-representation GGM proof at Appendix C.1.

Polynomial Representations As in the known-order GGM, we give the polynomial representations for each group element. However, as the group order is unknown, we choose the polynomials from $\mathbb{Z}[X_1, ..., X_t]$ *without modulus* for the formal variables $X_1, ..., X_t$ corresponding to the hidden values. In particular, if the algorithm takes no input, the representations could be in just $\mathbb{Z}$ without formal variables.

We extend the notion of informative collisions appropriately. We maintain the zero set $\mathcal{Z}$ and process each collision (i, j) with inputs corresponding to polynomials P_i, P_j (i.e., the equality gate outputting 1) as follows:

- If $P_i = P_j$ as a polynomial, then the collision is called *trivial*, and do nothing.
- If an equality query finds a nontrivial collision (i, j), then check if $P_i - P_j$ is included in $\mathbb{Z}$-span of $\mathcal{Z}$; recall the we only checked $\mathbb{Z}_N$-span in the known-order case. If it is not true, update $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{P_i - P_j\}$. We call the collision (i, j) *informative*, and otherwise *predictable.*

4.1 Order-finding in the Unknown-order GGM

We consider the following problem.

Problem 4. Let $\mathcal{D}^{(n)}_{\mathsf{prime}}$ be a uniform distribution over the set of n-bit primes. An order-finding problem over $\mathcal{D}^{(n)}_{\mathsf{prime}}$ in the GGM is defined as follows. First, a random N is sampled from $\mathcal{D}^{(n)}_{\mathsf{prime}}$. The adversary in the unknown-order GGM for the group $\mathcal{G}_N$ of order N is asked to output N. A product-order-finding problem over $\mathcal{D}^{(n)}_{\mathsf{prime}}$ is similarly defined, but the two distinct primes p, q are sampled and $N := pq$.

This problem is studied in [Sut07] in detail. In particular, the generic order-finding algorithm with the $O(\sqrt{N/\log\log N})$ group operation complexity suggested in [Sut07, Section 4], and the lower bound of $\Omega(N^{1/3})$ (for the prime-order case) is proven in the same thesis. We reprove this bound using our method.

[5] This was also used in the related works [DK02, Sut07] in the unknown-order GGM.

Theorem 4.1. *Let $\mathcal{A}_{\mathsf{ord}}$ be an order-finding algorithm over $\mathcal{D}^{(n)}_{\mathsf{prime}}$ in the GGM with the group operation complexity T. It holds that*

$$\Pr_{\mathcal{A}_{\mathsf{ord}},N}\left[\mathcal{A}^{\mathcal{G}_N}_{\mathsf{ord}}(g) \to N\right] = O\left(\frac{T^3}{2^n}\right).$$

In particular, any generic order-finding algorithm with a constant success probability must make $\Omega\left(N^{1/3}\right)$ group operations.

Proof. Let ϵ be the success probability of $\mathcal{A}_{\mathsf{ord}}$. For simplicity, we assume that $\mathcal{A}_{\mathsf{ord}}$ is deterministic.

We consider the integer representations corresponding to the elements of $\mathcal{A}_{\mathsf{ord}}$ because of the unknown order and no indeterminate value. In this case, an informative collision (i,j) must specify two integers x_i, x_j such that $x_i - x_j$ is a multiple of the order N. The integer x appearing in this list must satisfy

$$|x| \leq 2^T N$$

because a group operation only increases the number twice. This gives that the number of n-bit primes divisors of $x - y$ for some x, y appearing in the list is bounded by

$$\log_{2^n}(2^T N) = O\left(\frac{T}{\log N}\right).$$

We consider the following modification: If $\mathcal{A}_{\mathsf{ord}}$ outputs z, we let the algorithm make the labeling gate on inputs $z, 0$ and apply the equality gate on input (g^0, g^z) to find a collision at the end with probability at least ϵ.

Now, we construct the following encoding-decoding pair.

$\mathsf{Encode}(N)$**:** It runs $\mathcal{A}^{\mathcal{G}_N}_{\mathsf{ord}}(g)$ and computes the equality gate c with input (i,j) that is the lexicographically first informative collision. Let x_i, x_j be the corresponding integer representations. It factorizes $x_i - x_j$ and lets $p_1, ..., p_K$ be the n-bit prime divisors. It outputs (c, ℓ) where $p_\ell = N$ if exists. Otherwise, it outputs a special symbol $c = \perp$.

$\mathsf{Decode}(c, \ell)$**:** If $c = \perp$, it outputs a random sample from $\mathcal{D}^{(n)}_{\mathsf{prime}}$. Otherwise, it recovers x_i, x_j, computes and outputs the ℓ-th prime factor N'.

The correctness is analogous. The size of the encoding is $\log\binom{T}{2} + \log(K) + O(1)$, and we have $K = O(T/\log N)$. This gives

$$\log\binom{T}{2} + \log(K) + O(1) \geq \log\left(\frac{2^n}{n}\right) + O(1) + \log\epsilon \implies \epsilon = O\left(\frac{T^3}{N}\right)$$

applying Lemma 2.1 and $2^{n-1} \leq N \leq 2^n$. □

We can prove the analogous result for the product of two primes. The proof is essentially identical, except that we need to encode two prime factors using two informative collisions.

Theorem 4.2. *Let $\mathcal{A}_{\mathsf{ord}}$ be a product-order-finding algorithm over $\mathcal{D}^{(n)}_{\mathsf{prime}}$ in the GGM with the group operation complexity T. It holds that*

$$\Pr_{\mathcal{A}_{\mathsf{ord}},p,q}\left[\mathcal{A}^{\mathcal{G}_{pq}}_{\mathsf{ord}}(g) \to pq\right] = O\left(\frac{T^4}{2^{2n}}\right).$$

In particular, any generic order-finding algorithm with a constant success probability must make $\Omega\left(N^{1/4}\right)$ group operations for $N = pq$.

4.2 Root Extraction and Repeated Squaring Problems

We prove similar lower bounds for the (strong) root extraction and the repeated squaring in the unknown-order GGM.

Theorem 4.3. *Let $\mathcal{A}$ be an algorithm in the GGM with the group operation complexity T. Suppose that N is sampled from $\mathcal{D}^{(n)}_{\mathsf{prime}}$. It holds that*

$$\Pr_{\mathcal{A},N,x}\left[g^{ey} = g^x : \mathcal{A}^{\mathcal{G}_N}(g, g^x) \to (e, g^y)\right] = O\left(\frac{T^3}{2^n}\right).$$

Proof sketch. If there is an informative collision when running $\mathcal{A}$, we can apply the same encoding as in the previous section. We show that if the algorithm finds the root g^y, then it finds an informative collision.

Suppose that there is no informative collision during the execution for given input (g, g^x). Then, the polynomial corresponding to g^y must be $Y = aX + b \in \mathbb{Z}[X]$. The correctness implies that $eax + eb = x \bmod N$. To do so, either $N|ea - 1, N|eb$ or $x = eb/(ea - 1) \bmod N$ must hold. The first case implies that $N|b$ (otherwise $ea - 1$ is not divided by N), and since $|b|| \leq 2^T N$, this event only happens with probability at most $O(Tn/2^n)$. The second case holds with probability $1/N$. In other words, except this probability, the algorithm finds an informative collision. □

Theorem 4.4. *Let $\mathcal{A}$ be an algorithm in the GGM with the group operation complexity T. Suppose that N is sampled from $\mathcal{D}^{(n)}_{\mathsf{prime}}$. Let $t > T$ be a positive integer. It holds that*

$$\Pr_{\mathcal{A},N,x}\left[\mathcal{A}^{\mathcal{G}_N}(g) \to g^{2^t}\right] = O\left(\frac{(T+t)^3}{2^n}\right).$$

Proof sketch. If the algorithm $\mathcal{A}$ outputs h, we can compute g^{2^t} using t group operations and check if $h = g^{2^t}$. Also, the integer representation corresponding to h must be smaller than 2^T, thus it should be the informative collision. Using this, we can construct an encoding algorithm for N as in the previous section, proving the desired result. □

5 Lower Bounds in the Quantum GGM

5.1 Quantum Generic Group Models

5.1.1 Basic Quantum Generic Group Model

We define the quantum generic group model (QGGM) extending the model in Section 3.1, following the formalization in [HYY23]. Let $\mathcal{G}$ be a cyclic group of order N with a generator g. A quantum generic group algorithm $\mathcal{A}$ works similarly to a generic group algorithm but is defined on the registers holding qubits or superpositions of elements.

We first consider a rudimentary model, denoted by *the basic QGGM*, where group operations only work on two a priori fixed registers. As we look for the logarithmic lower bounds, we do not allow the quantum labeling gate and give a quantum inversion gate as a unit. An algorithm in the basic QGGM is defined as follows.

- There are two registers: qubit and element registers holding superpositions of some information. Qubit registers take a set of bits $\{0, 1\}$ as the computational basis. In contrast, element registers take a set of elements $x \in \mathcal{G} \cup \{\perp\}$ as the computational basis, which is denoted by g^x; sometimes $\perp$ is also written in this form though there is no corresponding x. The algorithm arbitrarily appends a new element register initialized by $|g\rangle$.
- There are (arbitrary) quantum gates that map qubits to qubits, which cannot take element registers as input.

- There are two special gates called *element gates* that can access the element wires as follows:

  Group Operation Gate. It takes two element registers $\mathbf{X}, \mathbf{Y}$ and a single qubit register $\mathbf{B}$ and applies the unitary $U_{\mathcal{G}.\mathsf{op}}$ that works on the computational basis as follows:

$$U_{\mathcal{G}.\mathsf{op}} : \begin{cases} |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} \mapsto |b\rangle_{\mathbf{B}} |g^{x+by}, g^y\rangle_{\mathbf{X},\mathbf{Y}} & \text{if } g^x, g^y \neq \perp, \\ |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} \mapsto |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} & \text{otherwise.} \end{cases} \tag{2}$$

  Inverse-Operation Gate. It takes two element registers $\mathbf{X}, \mathbf{Y}$ and a single qubit register $\mathbf{B}$ and applies the unitary $U_{\mathcal{G}.\mathsf{inv}}$ that works on the computational basis as follows:

$$U_{\mathcal{G}.\mathsf{inv}} : \begin{cases} |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} \mapsto |b\rangle_{\mathbf{B}} |g^{x-by}, g^y\rangle_{\mathbf{X},\mathbf{Y}} & \text{if } g^x, g^y \neq \perp, \\ |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} \mapsto |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} & \text{otherwise.} \end{cases}$$

  Equality Gate. It takes two element registers $\mathbf{X}, \mathbf{Y}$ and a single qubit register $\mathbf{B}$. It then applies the unitary operation $U_{\mathcal{G}.\mathsf{eq}}$ that works on the computational basis as follows:

$$U_{\mathcal{G}.\mathsf{eq}} : \begin{cases} |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} \mapsto |b \oplus \delta_{x,y}\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} & \text{if } g^x, g^y \neq \perp, \\ |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} \mapsto |b\rangle_{\mathbf{B}} |g^x, g^y\rangle_{\mathbf{X},\mathbf{Y}} & \text{otherwise,} \end{cases}$$

  where $\delta_{x,y} = 1$ if $x = y$ and 0 otherwise.

- We allow the intermediate measurements for registers. When we apply the measurements on all of $\mathbf{B}, \mathbf{X}, \mathbf{Y}$ right before applying the element gates, we call them *classical*. An element gate that is not classical is called quantum. For simplicity, we allow the classical labeling gate that does not much affect the result.

  Classical Labeling Gate. It takes $\lceil \log_2 N \rceil$ qubit registers, measures it, and interprets them as an element in $x \in \mathbb{Z}_N$. It appends a new element register holding $|g^x\rangle$. If there is no corresponding element $x \in \mathbb{Z}_N$ to the input wires, it outputs an element wire containing $|\perp\rangle$.

A QGGM algorithm denotes an algorithm $\mathcal{A}$ in this model and is occasionally written by $\mathcal{A}^{|\mathcal{G}\rangle}$. As in the classical GGM, we assume the element gates have some order to be applied sequentially with the relevant qubit gates.

The formal complexity measure of the generic algorithms is described in the next subsection. Roughly, we count the number of *quantum* element gates *including the equality gates* as the cost metric. The main reason is that while the equality check between two classical data is essentially free, e.g., using hash tables, the equality check between two element registers that store superpositions is not freely done. We also note that Shor's algorithm does not use any equality query.

5.1.2 QGGM with Coherent Indices

Now, we consider more general operations that can coherently access the indices of registers. Let t, w be positive integers. We define the (t, w)-QGGM similarly to the basic QGGM, but it also has *qudit* registers of dimension t and w, and the element gates are defined as follows.

- There are two special gates called *element gates* that can access the element wires as follows. The unspecified registers are unchanged by the operations.

  Group Operation Gate. It takes three registers $\mathbf{B}, \mathbf{T}, \mathbf{W}$ and $t+w$ element registers $\mathbf{X}_1, ..., \mathbf{X}_t, \mathbf{Y}_1, ..., \mathbf{Y}_w$ and applies the unitary $U^{(t,w)}_{\mathcal{G}.\mathsf{op}}$ that works on the computational basis as follows:

$$U^{(t,w)}_{\mathcal{G}.\mathsf{op}} : |b, i, j\rangle_{\mathbf{BTW}} |g^{x_i}, g^{y_j}\rangle_{\mathbf{X}_i,\mathbf{Y}_j} \mapsto |b, i, j\rangle_{\mathbf{BTW}} |g^{x_i+by_j}, g^{y_j}\rangle_{\mathbf{X}_i,\mathbf{Y}_j}$$

  for $i \in [t], j \in [w], g^{x_i}, g^{y_j} \neq \perp$ and $\mathbf{X}_i \neq \mathbf{Y}_j$, otherwise do nothing.

Inverse-Operation Gate. It takes three registers $\mathbf{B}, \mathbf{T}, \mathbf{W}$ and $t+w$ registers $\mathbf{X}_1, ..., \mathbf{X}_t, \mathbf{Y}_1, ..., \mathbf{Y}_w$ and applies the unitary $U_{\mathcal{G}.\mathsf{inv}}^{(t,w)}$ that works on the computational basis as follows:

$$U_{\mathcal{G}.\mathsf{inv}}^{(t,w)} : |b,i,j\rangle_{\mathbf{BTW}} |g^{x_i}, g^{y_j}\rangle_{\mathbf{X}_i,\mathbf{Y}_j} \mapsto |b,i,j\rangle_{\mathbf{BTW}} |g^{x_i - by_j}, g^{y_j}\rangle_{\mathbf{X}_i,\mathbf{Y}_j}$$

for $i \in [t], j \in [w], g^{x_i}, g^{y_j} \neq \perp$ and $\mathbf{X}_i \neq \mathbf{Y}_j$, otherwise do nothing.

Equality Gate. It takes three registers $\mathbf{B}, \mathbf{T}, \mathbf{W}$ and $t+w$ element registers $\mathbf{X}_1, ..., \mathbf{X}_t, \mathbf{Y}_1, ..., \mathbf{Y}_w$ and applies the unitary $U_{\mathcal{G}.\mathsf{eq}}^{(t,w)}$ that works on the computational basis as follows:

$$U_{\mathcal{G}.\mathsf{eq}}^{(t,w)} : |b,i,j\rangle_{\mathbf{BTW}} |g^{x_i}, g^{y_j}\rangle_{\mathbf{X}_i,\mathbf{Y}_j} \mapsto |b \oplus \delta_{x_i y_j}, i, j\rangle_{\mathbf{BTW}} |g^{x_i}, g^{y_j}\rangle_{\mathbf{X}_i,\mathbf{Y}_j}$$

for $i \in [t], j \in [w], g^{x_i}, g^{y_j} \neq \perp$ and $\mathbf{X}_i \neq \mathbf{Y}_j$, and do nothing other cases, where $\delta_{x,y} = 1$ if $x = y$ and 0 otherwise.

Note that the $(1,1)$-QGGM is identical to the basic QGGM. We remark that allowing coherent access to indices is relevant to practice. Coherent access to indices means the corresponding unitary operation should be large and implemented differently from the above gates. Furthermore, setting $t > 1$ implies that the quantum storage should store t group elements, requiring a large quantum memory. Allowing $w > 1$ was studied in [Gid19], and the estimation in [GE21] mainly used $t = 1$ and $w = 5$.

When we say *the QGGM*, the choice of (t, w) is unimportant in that context.

Remark 4. We did not explicitly state that the element registers are different. This potentially allows the group operations between the registers $\mathbf{X}_i, \mathbf{X}_j$.

5.1.3 Quantum Generic Group Algorithm with Classical Preprocessing

This paper considers the generic algorithms for the discrete logarithm that may perform classical generic computation before running the quantum parts. Formally, a generic (C, Q)-algorithm $\mathcal{A}$ in the QGGM decomposes into two generic algorithms $\mathcal{A}_c, \mathcal{A}_q$ as follows.

1. Given the problem instance, $\mathcal{A}_c$ consists of at most C classical group operation gates and arbitrarily many classical equality gates. It may have arbitrarily many qubit gates. At the end, it gives all registers to $\mathcal{A}_q$.
2. Given the registers from $\mathcal{A}_c$ as input, it applies at most Q quantum element gates along with arbitrarily many qubit gates. It measures the output registers and returns the measurement result as the outcome.

The following lemma shows the classical equality gates can be safely removed. The proof can be found in Appendix A.2.

Lemma 5.1. *Let $\mathcal{G}$ be a cyclic group of order N, and p the* smallest *prime divisor of N. For any $(C, 0)$-algorithm $\mathcal{A}_c$ in the (arbitrary) QGGM for $\mathcal{G}$, there is another $(C, 0)$-algorithm $\mathcal{A}'_c$ without equality gates such that*

$$\Pr_{\mathbf{x} \leftarrow [N]^m} [\mathcal{A}_c |0^n, g, g^{x_1}, ..., g^{x_m}\rangle = \mathcal{A}'_c |0^n, g, g^{x_1}, ..., g^{x_m}\rangle] \geq 1 - \frac{(C+m+1)^2}{2p}.$$

In particular, for generic (C, Q)-algorithms $\mathcal{A} = (\mathcal{A}_c, \mathcal{A}_q)$ and $\mathcal{A}' = (\mathcal{A}'_c, \mathcal{A}_q)$ for $\mathcal{A}'_c$ defined above, the outputs of two algorithms are identical with probability at least $1 - \frac{(C+m+1)^2}{2p}$.

5.2 Discrete Logarithms in the QGGM

The basic QGGM. We begin with the DL lower bound in the basic QGGM.

Theorem 5.2. *Let $\mathcal{G}$ be a cyclic group of order N with a generator g. Suppose that the smallest prime divisor of N is p. Let $\mathcal{A}_{\mathsf{DL}}$ be a (C,Q)-algorithm in the basic QGGM, then the following holds:*

$$\Pr_{\mathcal{A}_{\mathsf{DL}},x\leftarrow[N]}\left[\mathcal{A}_{\mathsf{DL}}^{|\mathcal{G}\rangle}(g,g^x)\to x\right]\leq\frac{(C+2)^2}{2p}+\frac{2^{2Q}}{N}.$$

Proof. Let ϵ be the success probability of $\mathcal{A}_{\mathsf{DL}}$. Decompose $\mathcal{A}_{\mathsf{DL}}=(\mathcal{A}_c,\mathcal{A}_q)$ as described in the previous section. By Lemma 5.1, it suffices to consider $\mathcal{A}'_{\mathsf{DL}}=(\mathcal{A}'_c,\mathcal{A}_q)$ where $\mathcal{A}'_c$ does not have any equality gates, whose output is identical to $\mathcal{A}_{\mathsf{DL}}=(\mathcal{A}_c,\mathcal{A}_q)$ with probability $1-(C+2)^2/2p$. In other words, $\mathcal{A}'_{\mathsf{DL}}$ solves the DL problem with a probability of at least

$$\epsilon'\geq\epsilon-\frac{(C+2)^2}{2p}. \tag{3}$$

Similarly to the classical case, we will construct an interactive compression protocol and apply Corollary 2.3. In the protocol, Alice holds group registers and applies element gates. Bob only holds the qubit registers and *delegates* all group-related operations to Alice.

We introduce the simple sub-protocols between Alice and Bob, showing that Alice sends one qubit during one element gate delegation.

Subprotocol Delegate.Gop **for group operation gates.** The initial states are

$$\sum_b\beta_b\left|b\right\rangle_{\mathbf{B}}\otimes\sum_{z,w}\alpha_{z,w}\left|g^z,g^w\right\rangle_{\mathbf{XY}} \tag{4}$$

where Alice holds the registers $\mathbf{X},\mathbf{Y}$ and Bob holds $\mathbf{B}$.

1. Bob sends his register $\mathbf{B}$ to Alice. Alice applies quantum group operation gates on $\mathbf{BXY}$ to obtain

$$\sum_b\beta_b\left|b\right\rangle_{\mathbf{B}}\otimes\sum_{z,w}\alpha_{z,w}\left|g^{z+bw},g^w\right\rangle_{\mathbf{XY}}. \tag{5}$$

2. Alice returns the register $\mathbf{B}$ to Bob.

In this protocol, Alice only sent one qubit and the group operation gate is applied as a result (compare Equations (4) and (5) and Equation (2)).

Subprotocol Delegate.Ginv **and** Delegate.Geq**.** These are almost the same as the protocol Delegate.Gop. The difference is as follows.

1. Alice applies the quantum inversion-operation gate or equality gate instead of the group operation gate.

Now, we return to the proof. We construct the following interactive protocol between Alice and Bob, where Alice selects $x\in[N]$ and tries to send x using this protocol. Given a (T,Q)-algorithm $\mathcal{A}'_{\mathsf{DL}}=(\mathcal{A}'_c,\mathcal{A}_q)$ for $T=C+2$, we consider the following protocol.

Main interactive protocol. Suppose that Alice chooses $x \in [N]$. In the protocol, Alice and Bob try to execute the algorithm $\mathcal{A}'_{\mathsf{DL}}$ together, while Alice holds all element registers and Bob holds all qubit registers. For qubit gates, Bob applies them locally without interacting with Alice.

To apply element gates, Alice and Bob use the above protocol. For the classical preprocessing, a simpler protocol suffices. We give the overall protocol below.

1. Alice prepares two element registers holding $|g, g^x\rangle$. If they are stored in the i, j-th element registers in $\mathcal{A}'_c$'s input, Alice also stores them in the i, j-th element registers.
2. Alice and Bob together execute $\mathcal{A}'_c$, with the following modifications.
   - Every qubit register is stored in Bob's memory, and every qubit gate is applied to Bob's side accordingly. Every element register is stored in Alice's memory. Alice and Bob use the same name/order of the registers as in $\mathcal{A}'_c$.
   - For each classical group operation gate that is applied to the registers $\mathbf{B}$ and $\mathbf{X}, \mathbf{Y}$, Bob measures $\mathbf{B}$ in the computational basis and sends the measurement outcome b to Alice. Alice applies the group operation on her registers $\mathbf{XY}$ controlled on b, and discards b.
   - Each classical labeling gate is processed analogously.

   We make some observations on this part. Alice's state is always classical during this procedure, so the measurement of Alice's registers can be ignored, and discarding bit b is not problematic. Alice has not sent any information to Bob until this point. Finally, the overall states between Alice and Bob are identical to the state after $\mathcal{A}'_c(g, g^x)$, except that all qubit registers are stored in Bob's memory and all element registers are stored in Alice's memory.
3. Alice and Bob execute $\mathcal{A}_q$ together in a similar way:
   - Every qubit gate is applied to Bob's registers accordingly.
   - For each group operation gate $U_{\mathcal{G}.\mathsf{op}}$ that is applied to the qubit register $\mathbf{B}$ and element registers $\mathbf{X}, \mathbf{Y}$, Alice and Bob executes $\mathsf{Delegate.Gop}$ on $\mathbf{BXY}$.
   - Similarly, $\mathsf{Delegate.Ginv}$ or $\mathsf{Delegate.Geq}$ is executed for each $U_{\mathcal{G}.\mathsf{inv}}$ or $U_{\mathcal{G}.\mathsf{eq}}$, respectively.
4. Finally, Bob outputs the final output of $\mathcal{A}_q$.

It is not hard to see that the overall states between Alice and Bob are always identical to the corresponding intermediate states of $\mathcal{A}'_{\mathsf{DL}}$ (ignoring the discarded bits). Therefore, the probability that Bob successfully recovers x is exactly the same as that $\mathcal{A}'_{\mathsf{DL}}$ solves the DL problem on input (g, g^x).

We then count the number of qubits sent from Alice to Bob. Alice sends a bit only when $\mathcal{A}_{\mathsf{DL}}$ applies an element gate. Thus, the total number of qubits is Q. At this point, we can apply Corollary 2.3 to have the following inequality:

$$\frac{\log \epsilon' + \log N}{2} \leq Q \implies \epsilon \leq \epsilon' + \frac{(C+2)^2}{2p} \leq \frac{2^{2Q}}{N} + \frac{(C+2)^2}{2p}$$

where we use Equation (3), which completes the proof. □

The (t, w)-QGGM. We then extend the lower bounds in the QGGM for more general settings. The proof ideas are almost the same, except for the sub-protocols; Alice needs to send one qudit for appropriate dimensions. We present the following generalization to the MDL problem.

Theorem 5.3. *Let $\mathcal{G}$ be a cyclic group of order N with a generator g. Suppose that the smallest prime divisor of N is p. Let m be a positive integer. Let $\mathcal{A}_{\mathsf{MDL}}$ be a (C, Q)-algorithm in the (t, w)-QGGM, then the following holds:*

$$\Pr_{\mathcal{A}_{\mathsf{DL}}, \mathbf{x} \leftarrow [N]^m} \left[\mathcal{A}_{\mathsf{DL}}^{|\mathcal{G}\rangle}(g, g^{\mathbf{x}}) \to \mathbf{x} \right] \leq \frac{(C+m+1)^2}{p} + \frac{(2tw)^{2Q}}{N^m}.$$

Proof. Applying Lemma 5.1, it suffices to consider the generic algorithm $\mathcal{A}'_{\mathsf{MDL}}$ with no classical equality queries, which solves the MDL problem with probability at least $\epsilon' \geq \epsilon - \frac{(C+m+1)^2}{2p}$. Then, we can construct a protocol between Alice and Bob where Alice aims to send $\mathbf{x} \in [N]^m$ to Bob using this algorithm. We need appropriate subroutines for the (t, w)-QGGM. For the group operation gate, it works as follows.

Subprotocol Delegate.Gop **for group operation gates.** The initial states are

$$\sum_{b,i\in[t],j\in[w]} \beta_{b,i,j} \left|b,i,j\right\rangle_{\mathbf{BTW}} \otimes \sum_{z,w} \alpha_{z,w} \left|...,g^z,...,g^w,...\right\rangle_{...\mathbf{X}_i...\mathbf{Y}_j...}$$

where Alice holds the registers $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_t), \mathbf{Y} = (\mathbf{Y}_1, ..., \mathbf{Y}_w)$ and Bob holds $\mathbf{B}, \mathbf{T}, \mathbf{W}$. g^z and g^w are stored in $\mathbf{X}_i, \mathbf{Y}_j$, respectively.

1. Bob sends $\mathbf{B}, \mathbf{T}, \mathbf{W}$ to Alice. Alice applies quantum group operation gates on $\mathbf{BTWXY}$.
2. Alice returns the register $\mathbf{B}, \mathbf{T}, \mathbf{W}$ to Bob.

In this protocol, Alice sends a quantum state of dimension $2tw$. The other element gates can be delegated analogously.

Each quantum element gate is operated with a quantum state with $2tw$ dimension, thus Corollary 2.3 implies that

$$\frac{\log \epsilon' + m \log N}{2} \leq Q \log(2tw) \implies \epsilon \leq \frac{(2tw)^{2Q}}{N^m} + \frac{(C+m+1)^2}{2p},$$

which concludes the proof. □

5.3 Unknown-order QGGM

We can prove the following QGGM variant of Theorems 4.1 and 4.2.

Theorem 5.4. *Let $\mathcal{A}_{\mathsf{ord}}$ be a (C, Q)-algorithm to solve the order-finding problem over $\mathcal{D}^{(n)}_{\mathsf{prime}}$ in the (t, w)-QGGM. It holds that*

$$\Pr_{\mathcal{A}_{\mathsf{ord}},N}\left[\mathcal{A}^{|\mathcal{G}_N\rangle}_{\mathsf{ord}}(g) \to N\right] = O\left(\frac{C^3}{2^n} + \frac{n(2tw)^{2Q}}{2^n}\right).$$

For the product-order-finding algorithm over $\mathcal{D}^{(n)}_{\mathsf{prime}}$ in the QGGM, it holds that

$$\Pr_{\mathcal{A}_{\mathsf{ord}},p,q}\left[\mathcal{A}^{|\mathcal{G}_{pq}\rangle}_{\mathsf{ord}}(g) \to pq\right] = O\left(\frac{T^4}{2^{2n}} + \frac{n^2(2tw)^{2Q}}{2^{2n}}\right).$$

Proof sketch. The proof is almost identical with the known-order QGGM proofs. Instead of applying Lemma 5.1, whenever the classical preprocessing finds an informative collision, we use it to compress N. Otherwise, Bob can delegate quantum group operations to Alice to construct the interactive protocol encoding N. The product-order-finding case is analogous. □

6 Lower Bounds in the Quantum Generic Ring Model

6.1 Quantum Generic Ring Model

We define the quantum generic group model (QGRM) in this section. The QGRM is a natural analog of the classical generic ring model [AM09, JS13], similar to the relation between the QGGM and GGM.

Let $\mathcal{R}$ be a commutative ring isomorphic to $\mathbb{Z}_N$ for some integer N to be specified later. Let t, w be positive integers. A quantum generic ring algorithm $\mathcal{A}$ in the QGRM is defined as follows. Note that the

definition of ring multiplication and division is rather complicated because of their subtlety; for example, they are not invertible as is, or there is no inverse. Our abstraction closely resembles the actual target arithmetic gates of circuit optimizations, e.g., [Bea03, Gid19]. We also remark that the (t, w)-QGGM can be defined analogously.

- There are two registers: qubit and element registers holding superpositions of some information. In contrast, element registers take a set of elements $x \in \mathcal{R} \cup \{\perp\}$ as the computational basis. The algorithm arbitrarily appends a new element register initialized by $|0\rangle$ or $|1\rangle$.
- There are (arbitrary) quantum gates that map qubits to qubits, which cannot take element registers as input.
- There are special gates called *element gates* that can access the element wires as follows. The unspecified registers are unchanged by the operations.

  Ring Addition Gate. It takes a qubit register $\mathbf{B}$ and two element registers $\mathbf{X}, \mathbf{Y}$ and applies the unitary that works on the computational basis as follows:

  $$|b\rangle_{\mathbf{B}} |x, y\rangle_{\mathbf{XY}} \mapsto |b\rangle_{\mathbf{B}} |x + by, y\rangle_{\mathbf{XY}}$$

  for $x, y \neq \perp$, otherwise do nothing.

  Ring Subtraction Gate. It is essentially identical to the ring addition gate, except for the choice of unitary:

  $$|b\rangle_{\mathbf{B}} |x, y\rangle_{\mathbf{XY}} \mapsto |b\rangle_{\mathbf{B}} |x - by, y\rangle_{\mathbf{XY}} .$$

  Ring Product-Addition Gate. It takes a qubit register $\mathbf{B}$ and three element registers $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and applies the unitary that works on the computational basis as follows:

  $$|b\rangle_{\mathbf{B}} |x, y, z\rangle_{\mathbf{XYZ}} \mapsto |b\rangle_{\mathbf{B}} |x + byz, y, z\rangle_{\mathbf{XYZ}}$$

  for $x, y, z \neq \perp$ and registers, otherwise do nothing.

  Testing Invertible Gate. It takes an element register $\mathbf{X}$ and a qubit register $\mathbf{C}$, and applies the unitary U_{Test} that works on the computational basis as follows:

  $$U_{\mathsf{Test}} : |x, c\rangle_{\mathbf{XC}} \mapsto |x, c \oplus \mathsf{Test}(x)\rangle_{\mathbf{XC}}$$

  where $\mathsf{Test}(x) = 1$ if x is invertible, otherwise $\mathsf{Test}(x) = 0$.

  Ring inversion-addition Gate. It takes a qubit register $\mathbf{B}$ and three element registers $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$. It appends an ancillary qubit register $\mathbf{C}$ initialized by $|0\rangle$ and applies the following sequence of unitaries that works on the computational basis as follows:

  $$\begin{aligned}
  &|b\rangle_{\mathbf{B}} |x, y, z\rangle_{\mathbf{XYZ}} |0\rangle_{\mathbf{C}} \\
  &\mapsto |b\rangle_{\mathbf{B}} |x, y, z\rangle_{\mathbf{X}_i, \mathbf{Y}_j, \mathbf{Y}_k} |\mathsf{Test}(z)\rangle_{\mathbf{C}} \\
  &\mapsto |b\rangle_{\mathbf{B}} \left|x, +b\mathsf{Test}(z) y \cdot z^{-1}, y, z\right\rangle_{\mathbf{XYZ}} |\mathsf{Test}(z)\rangle_{\mathbf{C}} \\
  &\mapsto |b\rangle_{\mathbf{B}} \left|x, +b\mathsf{Test}(z) y \cdot z^{-1}, y, z\right\rangle_{\mathbf{XYZ}} |0\rangle_{\mathbf{C}}
  \end{aligned}$$

  for $x, y, z \neq \perp$, and do nothing other cases. Here, the first and last unitaries are U_{Test} on $\mathbf{ZC}$. In the second unitary, it adds $y \cdot z^{-1}$ only if $\mathsf{Test}(y_k) = 1$ and $b = 1$. It discards $\mathbf{C}$ in the end.

  Equality Gate. It takes a qubit register $\mathbf{B}$ and two element registers $\mathbf{X}, \mathbf{Y}$ and applies the unitary that works on the computational basis as follows:

  $$|b\rangle_{\mathbf{B}} |x, y\rangle_{\mathbf{XY}} \mapsto |b \oplus \delta_{x,y}\rangle_{\mathbf{B}} |x, y\rangle_{\mathbf{XY}}$$

  for $x, y \neq \perp$, otherwise do nothing, where $\delta_{x,y} = 1$ if $x = y$ and 0 otherwise.

- We allow the intermediate measurements for registers and the labeling gate.

  Classical Labeling Gate. It takes $\lceil \log_2 N \rceil$ qubit registers, measures it, and interprets them as an element in $x \in \mathbb{Z}_N \simeq \mathcal{R}$. It appends a new element register holding $|x\rangle$. If there is no corresponding element $x \in \mathbb{Z}_N$ to the input wires, it outputs an element wire containing $|\perp\rangle$.

We count the number of element gates as the cost metric, denoted by *the ring operation complexity*.

We stress that the modulus N is *NOT* given to the generic algorithms explicitly. Instead, the algorithm only accesses the ring (or modular) operations. This still captures most quantum parts of the quantum factoring algorithms. For Shor's algorithm, the quantum part aims to find the order r of the randomly chosen integer a, which only requires modular arithmetic. We prove the lower bound for the order finding in the QGRM in Theorem 6.1. The knowledge of N beyond our model is used in the classical post-processing parts for computing (say) $\gcd(a^{r/2}+1, N)$.

Regev's algorithm does not compute the order. Instead, it computes a short vector $\mathbf{z} = (z_1, ..., z_d)$ in a certain lattice, and then compute $\gcd(b_1^{z_1}...b_d^{z_d} - 1, N)$. In the QGRM, the algorithm still can compute $b_1^{z_1}...b_d^{z_d} - 1$ as an integer, which is relatively small, having a nontrivial common factor with N. We prove that this algorithm needs to make a logarithmic number of ring operations in Theorem 6.2.

6.1.1 Quantum Generic Ring Algorithm with Classical Preprocessing

We consider a slightly more general algorithm that can do classical preprocessing *without* the testing or equality gates. The classical ring operations are defined by the ring operations that element gates are measured before applying ring operations. Recall the ring operations are done on Alice's side in the proofs. The delegation of classical ring operations can be done without Alice's messages, except for the equality gates, where Bob needs to take the output of gates. Therefore, the number of classical ring operations, such as precomputing x^{2^k}, is irrelevant to our lower bounds. We do not explicitly discuss this setting in the remainder of this section.

6.2 Lower Bounds in the QGRM

This section is devoted to proving that the order-finding problems with certain distributions and a certain type of factoring algorithms have logarithmic ring operation complexity. In a ring $\mathcal{R} \simeq \mathbb{Z}_N$, the order of $x \in \mathcal{R}$, denoted by $\mathsf{ord}_N(x)$, is defined by the minimal positive integer e such that $x^e = 1 \bmod N$. A prime number p is *safe* if $\frac{p-1}{2}$ is also prime. We consider the following problem.

Problem 5. Let $\mathcal{D}_{\mathsf{safe}}^{(n)}$ be a uniform distribution over the set of n-bit safe primes. An order-finding problem over $\mathcal{D}_{\mathsf{safe}}^{(n)}$ in the QGRM is defined as follows. First, two distinct primes p, q are sampled from $\mathcal{D}_{\mathsf{safe}}^{(n)}$ and let $N = pq$. Choose a random $x \in \mathbb{Z}_N$. The adversary is given x stored in the element register in the QGRM for $\mathcal{R} \simeq \mathbb{Z}_N$ and asked to find $\mathsf{ord}_N(x)$.

We assume that the number of n-bit safe primes is at least $C \cdot 2^n/n^2$ for some constant $C > 0$, which is a variant of the conjecture that the number of safe primes below N is of order $\Theta(N/\log^2 N)$ [Sho09, Section 5.5.5].

Theorem 6.1. *Let $\mathcal{A}_{\mathsf{ord}}$ be an order-finding algorithm over $\mathcal{D}_{\mathsf{safe}}^{(n)}$ in the (t, w)-QGRM with the ring operation complexity of Q. Assuming that the number of n-bit safe primes is at least $C \cdot 2^n/n^2$ for $C > 0$, it holds that*

$$\Pr_{\mathcal{A}_{\mathsf{ord}}, p, q, x}[\mathcal{A}_{\mathsf{ord}}(x) \to \mathsf{ord}_{pq}(x)] = O\left(\frac{n^4(2tw)^{2Q}}{2^{2n}} + \frac{1}{2^n}\right)$$

Proof. Let $N = pq$. We first observe that the order of x is a divisor of $\frac{(p-1)(q-1)}{2} = 2 \cdot \frac{p-1}{2} \cdot \frac{q-1}{2}$. With probability $1 - O(1/p)$ over random x, $\mathsf{ord}_N(x) = \frac{(p-1)(q-1)}{2}$ or $\frac{(p-1)(q-1)}{4}$. In this case, one can recover (p, q) for safe primes p, q from $\mathsf{ord}_N(x)$ using the factorization.

Based on this observation, we construct a protocol between Alice and Bob where Alice wants to send (p, q) to Bob. The proof is identical to that of Theorem 5.2, except that we need the delegation sub-protocols for ring operations. By the assumption, Alice sends one out of $O\left(\frac{2^{2n}}{n^4}\right)$ candidates to Bob using Q qubits of communications. □

We then consider the factoring algorithms, where the generic algorithm's goal is to find an integer with a nontrivial common divisor with N. We prove the following theorem.

Theorem 6.2. *Let $\mathcal{A}$ be an algorithm in the (t, w)-QGRM with the ring operation complexity of Q. For two primes p, q sampled from $\mathcal{D}^{(n)}_{\mathsf{prime}}$ and $N = pq$, it holds that*

$$\Pr_{\mathcal{A},p,q}[1 < \gcd(Z, N) < N : \mathcal{A}() \to Z] = O\left(\frac{n \log Z (2tw)^{2Q}}{2^n}\right).$$

In particular, if $\log Z = O(2^{(2-\epsilon)n})$ for any $\epsilon > 0$, this implies that $Q = \Omega(\frac{\log N}{\log(2tw)})$ to have the constant success probability.

Before proceeding with the proof, we give some interpretations of this theorem. As we do *not* give N to the generic algorithm, it cannot apply the modulus operation. Therefore, the known quantum factoring algorithms must be explained with some modifications, where the final steps usually compute the common divisor of some integer and N.

Instead of giving N, we ask to find an integer that suffices for factoring N. In the QGRM, this integer must be computed in plain, without modulus computation. For Regev's algorithm, the final integer is of the form $Z = \prod_{i\in[d]} b_i^{z_i}$ for $z_i = \exp(O(\sqrt{n}))$ and $d \approx \sqrt{n}$. The last statement holds in this case as well.

The output of Shor's algorithm corresponds to $Z = a^{r/2} - 1$ for $r = \mathsf{ord}_N(a)$. The bit length of Z is about $\log Z \leq \frac{r}{2} \cdot \log a \leq \frac{nN}{2}$. Therefore, we cannot apply this theorem to Shor's algorithm in general.

Proof. Let ϵ be the success probability of $\mathcal{A}$. We construct a protocol that sends (p, q) using $\mathcal{A}$. Precisely, Bob runs $\mathcal{A}$ using the delegation of quantum ring operations using $Q \log(2tw)$ qubits. After obtaining the outputs Z from $\mathcal{A}$, Alice additionally sends an index of the prime factor of Z among its n-bit prime factors. Since the number of n-bit primes factors of Z is bounded by $\log Z/n$, the index can be described in $\log(\log Z) - \log n$ classical bits. Finally, Alice sends the other prime factor which can be specified by $n - \log n + O(1)$ classical bits. Appying Corollary 2.3, we have

$$2Q \log(2tw) + (\log\log Z - \log n) + (n - \log n + O(1)) \geq 2n - 2\log n + \log\epsilon + O(1),$$

which implies

$$\epsilon = O\left(\frac{n(\log Z)(2tw)^{2Q}}{2^n}\right),$$

concluding the proof. □

7 Lower Bounds for Index Calculus Algorithms

This section introduces a new model called *the smooth index calculus model (SGGM)* of generic algorithms, including the (simplest) index calculus methods.

A main feature of index calculus is using the set of B-smooth numbers, denoted by S_B, whose prime factors are all less than or equal to B. These numbers are relatively quickly factorized, and the index calculus method finds many nontrivial elements in S_B to leverage this fact.

7.1 Smooth Generic Group Model

The smooth GGM is parameterized by a parameter B, which induces the factor base $\mathcal{B}$ and the set $S = \{h_1, ..., h_{|S|}\}$ of smooth elements. Precisely, the factor base is a set of primes $\mathcal{B} = \{p_1, ..., p_b\}$ and the smooth element $h_i \in S$ is of the form

$$h_i = p_1^{c_1^{(i)}} \cdot \cdots \cdot p_b^{c_b^{(i)}} \tag{6}$$

for $\mathbf{c}^{(i)} = (c_1^{(i)}, ..., c_b^{(i)})$, whose precise conditions will be specified later.

An SGGM algorithm $\mathcal{A}$ over $\mathcal{G}$ of order N with the parameter B, denoted by an algorithm in the B-SGGM, is given by a circuit with the following features:

- There are two types of wires: bit wires and element wires. Bit wires take a bit in $\{0, 1\}$, and element wires take an element in $x \in \mathbb{Z}_N \cup \{\perp\}$, which is denoted by g^x.
- There are bit gates and element gates that are identically defined as the generic group model.
- There are special element gates defined as follows:

  Smooth Test Gate. It takes an element wire containing h. If $h \in S$, it outputs 1, otherwise outputs 0.

  Smoothing Gate. It takes an element wire containing h. If it is smooth, i.e., $h = h_i$ for some $i \in [|S|]$, outputs $\mathbf{c}^{(i)}$ defined in Equation (6). Otherwise, it outputs $\perp$.

We further establish the properties of the factor base and the smooth elements regarding the parameter B. Let g be the generator of $\mathcal{G}$, and let $u > 0$ be such that $B = N^{1/u}$. Let $c_{\mathsf{base}}, c_{\mathsf{smooth}}, d_{\mathsf{smooth}} \geq 1$ be the universal constants that are independent from B. Here, $o(1)$ hides a factor much less than 1.

- The set $\mathcal{B}$ is given to the algorithm. The set S is randomly chosen and unknown to the algorithm. For the factor base $\mathcal{B} = \{p_1, ..., p_b\}$, it holds that $p_i = g^{z_i}$ for some *random* z_i for each i, which is unknown to the algorithm.
- The size of the factor base $|\mathcal{B}| = b$ is $(c_{\mathsf{base}} + o(1))B/\log B$.
- The number of smooth elements is

$$p_S := \frac{|S|}{N} = (c_{\mathsf{smooth}} + o(1)) \cdot \left(\frac{d_{\mathsf{smooth}} + o(1)}{u \log u}\right)^u. \tag{7}$$

- For any rank-c affine space V in $\mathbb{Z}_N^b$, define

$$S_V := \{h^{(i)} \in S : \mathbf{c}^{(i)} \in V\}.$$

  If $c = (c_{\mathsf{base}} + o(1))C/\log C$ for some $C = N^{1/v}$, it holds that

$$\frac{|S_V|}{N} \leq (c_{\mathsf{smooth}} + o(1)) \cdot \left(\frac{d_{\mathsf{smooth}} + o(1)}{v \log v}\right)^v. \tag{8}$$

We explain the reasoning behind these assumptions. The assumption on the prior knowledge of the algorithm reflects the reality. The randomness of S and z_i prevents the generic algorithm from using the explicit values related to the smooth elements.

The sizes of $\mathcal{B}$ and S stem from the original choices in the index calculus, whose estimated sizes are well-studied. We refer the survey on this topic [Gra08] to the readers.

The last assumption describes that the vectors $\mathbf{c}^{(i)}$ are *well-distributed*. In particular, it asserts that the factor base $\{p \leq C\}$, which corresponds to the subspace $V = \mathbb{Z}_N^c \times \{0\}^{b-c}$, maximizes the size of S_V, according to the estimated size by Equation (7).

7.1.1 Polynomial Representations

Again, we identify the group elements by the corresponding polynomials. We mainly focus on the discrete logarithm problems where the problem instance is given as $(g, g^{x_1}, ..., g^{x_m})$, which corresponds to $1, X_1, ..., X_m$. Furthermore, because of the factor base, we have more formal variables $Z_1, ..., Z_b$. Therefore, each element corresponds to the polynomial in

$$\mathbb{Z}_N[X_1, ..., X_m, Z_1, ..., Z_b].$$

We stress that we occasionally identify the polynomial with its coefficient vector.

We consider the answer from the smoothing gate to be the collision. Precisely, if an element h corresponding to the polynomial P is given to the smoothing gate and the answer is $(c_1, ..., c_b)$, then it induces the collision

$$P = c_1 Z_1 + ... + c_b Z_b.$$

If it is not included in the span of the previous zero set $\mathcal{Z}$, we include it as an informative collision as well.

7.2 The Discrete Logarithm Problem in the SGGM

In this section, we assume that the variables $N = N(\lambda), B = B(\lambda), u = u(\lambda)$ are parameterized by some implicit parameter λ so that we can work in the asymptotic regime. Still, we drop the parameter λ for simplicity.

Theorem 7.1. *Let $\mathcal{G}$ be a cyclic group of prime order N. Let B be an integer such that $B = N^{1/u}$ for some $u > 0$. Let $\mathcal{A}_{\mathsf{DL}}$ be a DL algorithm in the B-SGGM with a constant success probability. Then, the number of group operations T of $\mathcal{A}_{\mathsf{DL}}$ must satisfy*

$$T = \exp\left(\Omega\left(\sqrt{\log N \log\log N}\right)\right).$$

Proof. Toward contradiction, we assume that $\mathcal{A}_{\mathsf{DL}}$ successfully solves the DL problem in the SGGM with smaller group operations than the statement. We first observe that each equality query makes an informative collision with probability $\frac{1}{N}$; thus, with probability $1 - \frac{T^2}{N}$, there are no informative collisions from the equality queries. From now on, we ignore the equality gates and assume that all informative collisions are from the smoothing gate.

As before, we assume that $\mathcal{A}_{\mathsf{DL}}$ makes the equality gate at the end so that the collision is found. We begin with the following fact, which is a SGGM variant of Lemma 3.2.

Fact 3. Each group operation introduces a new informative collision (through the smoothing gate) with probability at most p_S defined in Equation (7). In particular, the input element to the smoothing gate collides with a random element in S.

Proof of fact. Let h be a new group element corresponding to $(c_0, a, c_1, ..., c_b)$, which is linearly independent from the vectors in $\mathcal{Z}$. This means that h is uniformly distributed over random $X, Z_1, ..., Z_b$ conditioned on the equations in $\mathcal{Z}$ hold. That is, $h \in S$ holds with probability $|S|/N = p_S$. □

Case 1. We first consider the case that u is sufficiently large so that

$$u^{8u^2} \geq N \implies u \log u = \Omega\left(\sqrt{\log N \log\log N}\right).$$

In this case, by Fact 3, $\mathcal{A}_{\mathsf{DL}}$ must make

$$1/p_S = \Omega(u^u) = \exp\left(\Omega\left(\sqrt{\log N \log\log N}\right)\right)$$

group operations to find an informative collision with a constant probability.[6]

[6] A formal proof requires some probabilistic arguments, which we omitted here.

Case 2. We consider the other case that u is relatively small so that

$$u^{8u^2} \leq N.$$

In this case, we choose $v > u$ such that $v^{v^2} = N$. Note that

$$(2u)^{(2u)^2} = (2u)^{4u^2} \leq u^{8u^2} \leq N,$$

thus $v \geq 2u$ holds. Suppose that the algorithm finds K informative collisions at total. We will prove that $K = \Omega\left(N^{1/2v}\right)$ in this case. Since

$$v^2 \log v = \log N \implies v = \Theta\left(\sqrt{\frac{\log N}{\log\log N}}\right),$$

we have

$$T \geq K = \exp\left(\Omega\left(\frac{\log N}{v}\right)\right) = \exp\left(\Omega\left(\sqrt{\log N \log\log N}\right)\right).$$

Combining the two cases, we prove the theorem.

It remains to prove the lower bound of K in the second case. We identify the formal variable X to represent g^x. In particular, the span of the final zero set $\mathcal{Z}$ must include the polynomial $X - x$, or a vector $(-x, 1, 0, ..., 0)$. We define $\mathcal{Z}^{(t)}$ to denote the zero set right after the t-th informative collision. Define the following projections of $\mathcal{Z}^{(t)}$:

$$\mathcal{Z}^{(t)}_{X=x} := \{(ax + c_0, c_1, ..., c_b) : c_0 + aX + c_1 Z_1 + ... + c_b Z_b \in \mathcal{Z}^{(t)}\}$$
$$\mathcal{Z}^{(t)}_{\mathcal{B}} := \{(c_1, ..., c_b) : c_0 + aX + c_1 Z_1 + ... + c_b Z_b \in \mathcal{Z}^{(t)}\}$$

We observe the following facts.

Fact 4. The rank of $\mathcal{Z}^{(K)}_{X=x}$ is less than K. In particular, there must exist a smoothing gate making the $t(\leq K)$-th informative collision P such that $P^{(t)}_{X=x}$ is included in the span of $\mathcal{Z}^{(t-1)}_{X=x}$. The rank of $\mathcal{Z}^{(t)}_{X=x}$ is equal to the rank of $\mathcal{Z}^{(t)}_{\mathcal{B}}$ for each $t \in [K]$.

Proof of fact. Let $(-x, 1, 0, ..., 0) = \mathbf{b}$ and $\{\mathbf{b}, \mathbf{b}_2, ..., \mathbf{b}_K\}$ be the basis extension of $\mathcal{Z} = \mathcal{Z}^{(K)}$ from $\{\mathbf{b}\}$. The projection $\pi : (c_0, a, c_1, ..., c_b) \mapsto (c_0 + ax, c_1, ..., c_b)$ maps $\mathcal{Z}$ to $\mathcal{Z}^{(K)}_{\mathcal{B}}$ and $\pi(\mathbf{b}) = 0$, thus the rank of $\mathcal{Z}^{(K)}_{\mathcal{B}}$ must be $K - 1$. The final statement follows from $(1, 0, ..., 0)$ is not included in the span of $\mathcal{Z}$. □

We call the first smoothing gate by *critical* with input h and output $h^{(i)} \in S$ satisfying the condition described in Fact 4. Let $(h_1, ..., h_b)$ and $\mathbf{c}^{(i)}$ be the corresponding coefficient vectors of h and $h^{(i)}$.

We give an upper bound for the probability p_t that the t-th informative collision is critical. This means that the rank of $\mathcal{Z}^{(t-1)}_{\mathcal{B}}$ is $t - 1$, and $(h_1, ..., h_b) - \mathbf{c}^{(i)}$ is included in $\mathcal{Z}^{(t-1)}_{\mathcal{B}}$. In other words,

$$\mathbf{c}^{(i)} \in (h_1, ..., h_b) + \mathsf{span}\left(\mathcal{Z}^{(t-1)}_{\mathcal{B}}\right) =: V.$$

Since $h^{(i)}$ is a random element in S, Equation (8) implies that the probability p_t is bounded by

$$p_t = \Pr\left[\mathbf{c}^{(i)} \in V\right] = \frac{|S_V|}{|S|}.$$

Let $C = N^{1/v}$ and $c = c_{\mathsf{base}} C / \log C$. If $t \leq c$, the logarithm of the above equation becomes for a constant $\alpha \approx \log d_{\mathsf{smooth}}$

$$\log\left(\frac{|S_V|}{|S|}\right) \leq -v\log(v\log v) + u\log(u\log u) + \alpha(v-u) + O(1)$$
$$\leq -0.5v\log(v\log v) - u\log(2u\log 2u) + u\log(u\log u) + \alpha v + O(1)$$
$$\leq -0.5v\log v + O(1).$$

where we use the fact that d_{smooth} is constant and $v \geq 2u$, and set $\alpha \approx \log d_{\mathsf{smooth}}$, which is less than $0.5 + 0.5 \log \log v$ in the interested parameter regime. It implies that $p_t \leq \beta / v^{0.5v}$ for some constant $\beta > 0$. In other words, with constant probability, the critical informative collision will be found after $\Omega(v^{0.5v}) = \Omega(N^{1/2v})$ informative collisions are found. □

References

[AGK20] Benedikt Auerbach, Federico Giacon, and Eike Kiltz. Everybody's a target: Scalability in public-key encryption. 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 III*, volume 12107 of *Lecture Notes in Computer Science*, pages 475–506. Springer, 2020.

[AHP23] Benedikt Auerbach, Charlotte Hoffmann, and Guillermo Pascual-Perez. Generic-group lower bounds via reductions between geometric-search problems: With and without preprocessing. *IACR Cryptol. ePrint Arch.*, page 808, 2023.

[AM09] Divesh Aggarwal and Ueli Maurer. Breaking rsa generically is equivalent to factoring. In *Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28*, pages 36–53. Springer, 2009.

[Bea03] Stephane Beauregard. Circuit for shor's algorithm using 2n+ 3 qubits. *Quantum Information & Computation*, 3(2):175–185, 2003.

[BFP21] Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. The one-more discrete logarithm assumption in the generic group model. 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 IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 587–617. Springer, 2021.

[BL96] Dan Boneh and Richard J Lipton. Algorithms for black-box fields and their application to cryptography. In *Annual International Cryptology Conference*, pages 283–297. Springer, 1996.

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking rsa may be easier than factoring. In *Advances in Cryptology—EUROCRYPT*, volume 98, pages 59–71. Citeseer, 1998.

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 693–721. Springer, 2018.

[CK18] Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 415–447. Springer, 2018.

[CLQ19] Kai-Min Chung, Tai-Ning Liao, and Luowen Qian. Lower bounds for function inversion with quantum advice. *arXiv preprint arXiv:1911.09176*, 2019.

[Den02] Alexander W Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 100–109. Springer, 2002.

[DGK17] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 473–495. Springer, 2017.

[DK02] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 256–271. Springer, 2002.

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In *Annual Cryptology Conference*, pages 649–665. Springer, 2010.

[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.

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

[Gra08] Andrew Granville. Smooth numbers: computational number theory and beyond. *Algorithmic number theory: lattices, number fields, curves and cryptography*, 44:267–323, 2008.

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 305–313. IEEE, 2000.

[HNR18] Shima Bab Hadiashar, Ashwin Nayak, and Renato Renner. Communication complexity of one-shot remote state preparation. *IEEE Transactions on Information Theory*, 64(7):4709–4728, 2018.

[HXY19] Minki Hhan, Keita Xagawa, and Takashi Yamakawa. Quantum random oracle model with auxiliary input. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 584–614. Springer, 2019.

[HYY23] Minki Hhan, Takashi Yamakawa, and Aaram Yun. Quantum complexity for discrete logarithms and related problems. *arXiv preprint arXiv:2307.03065*, 2023.

[JS13] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. *Journal of cryptology*, 26:225–245, 2013.

[KM06] Neal Koblitz and Alfred Menezes. Another look at generic groups. *Cryptology ePrint Archive*, 2006.

[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.

[NABT15] Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. Quantum lower bound for inverting a permutation with advice. *Quantum Information & Computation*, 15(11-12):901–913, 2015.

[Nec94] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[NS06] Ashwin Nayak and Julia Salzman. Limits on the ability of quantum states to convey classical messages. *Journal of the ACM (JACM)*, 53(1):184–206, 2006.

[PH78] S Pohlig and M Hellman. An improved algorithm for computing logarithms over gf (p) and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[Pol78] John M Pollard. Monte carlo methods for index computation (mod p)). *Mathematics of computation*, 32(143):918–924, 1978.

[Reg23] Oded Regev. An efficient quantum factoring algorithm. *arXiv preprint arXiv:2308.06572*, 2023.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[Sut07] Andrew V Sutherland. *Order computations in generic groups*. PhD thesis, Massachusetts Institute of Technology, 2007.

[Yun15] Aaram Yun. Generic hardness of the multiple discrete logarithm problem. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 817–836. Springer, 2015.

[YYHK20] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. Generic hardness of inversion on ring and its relation to self-bilinear map. *Theoretical Computer Science*, 820:60–84, 2020.

[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.

A Missing Proofs

A.1 Missing proofs in the GGM

Proof of Lemma 3.1. Suppose that $\mathcal{A}$ is deterministic; if $\mathcal{A}$ is randomized, we include the random seed as its description. We construct an algorithm $\mathcal{A}'$ that has the same circuits as $\mathcal{A}$, but the element wires are replaced by the polynomial wires. The initialization $\mathcal{P} = \{(w_1, P_1), ..., (w_m, P_m)\}$ for input can be done without any query, and each input element wire w_i is replaced by P_i in $\mathcal{A}'$. The labeling gates and group operation gates are processed as in the polynomial list. For the equality gates with element wires w_i, w_j, the output can be computed by checking if $P_i - P_j$ is included in the span of $\mathcal{Z}$; if included the output of the equality gate is 1, otherwise 0. □

Proof of Fact 1. Let $P_i := a_{i,1}X_1 + ... + a_{i,m}X_m + b_i$. A collision (i, j) induces a linear equation over $\mathbb{Z}_p$ as

$$0 = P_i - P_j = (a_{i,1} - a_{j,1})X_1 + ... + (a_{i,m} - a_{j,m})X_m + (b_i - b_j).$$

Since collisions are informative, they are nontrivial and linearly independent due to Equation (1). Thus m informative collisions give a system of m linear equations over $\mathbb{Z}_p$ with m variables that are linearly independent, which can be easily solvable. □

Proof of Theorem 3.6, sketch. Observe that without finding an informative collision, the output should correspond to $aX + bY + c$ for some a, b, c, where X, Y are the variables corresponding to g^x, g^y. The probability that $aX + bY + c = XY$ is at most $1/|\mathcal{G}|$ over the random choice of X, Y. Therefore, the algorithm must find an informative collision.

Given an informative collision exists, the first part of the encoding is the first informative collision. If this collision has a nonzero coefficient for the monomial containing X, then the second part of the encoding is x. Otherwise, it is y. Given the encoding, the decoding procedure is 1) parses the first informative collision and one of x or y, and 2) plugs it in the first collision. The collision collapses to a one-variable polynomial of degree less than 1, and the correct solution can be guessed with probability at least $1/2$. □

A.2 A QGGM lemma

Proof of Lemma 5.1. Except for the equality gates, we define the algorithm $\mathcal{A}'_c$ as identical to $\mathcal{A}_c$. It removes all equality gates, except the trivial equality gate (as in the classical GGM) that are replaced by bit flipping. Given the first assertion, the "In particular" part is obvious because the trace distance between the intermediate outputs of two (C, Q)-algorithms are identical with probability $1 - \frac{(C+m+1)^2}{2p}$, and the remaining parts are the same.

The proof proceeds as follows. As the algorithm $\mathcal{A}_c$ is only given classical group elements and can apply classical group gates, it only maintains at most $C+m+1$ classical group elements, which are represented by polynomials $P_1, ..., P_{C+m+1}$ as done in the classical generic group models. Each equality query corresponds to the difference between polynomials $P_i - P_j$. If $P_i - P_j$ is identically zero or never be zero, then there is no difference between $\mathcal{A}_c$ and $\mathcal{A}'_c$ from these equality gates.

Consider an equality gate corresponding to $P_i - P_j$ that is not identically zero. There exists some prime power q^t that exactly divides N such that $P_i - P_j$ is nonzero modulo q^t. Since $P_i - P_j$ is linear, the portion of inputs where the equality gate corresponding to $P_i - P_j$ behaves differently from the identity gate is at most $1/q \leq 1/p$. Since there are at most $\binom{C+m+1}{2} \leq \frac{(C+m+1)^2}{2}$ different pairs of group elements, at most $\frac{(C+m+1)^2}{2p}$-fraction of inputs make difference on the behaviors of $\mathcal{A}_c$ and $\mathcal{A}'_c$. In other words, the output states of the two algorithms are identical with probability at least $1 - \frac{(C+m+1)^2}{2p}$ for random inputs. □

B An Alternative Proof for the MDL Lower Bound

We give a simple proof for the MDL lower bound Theorem 3.4. We begin with the proof of Lemma 3.2.

Proof of Lemma 3.2. Let $\mathcal{Z} = \{Q_1, ..., Q_s\}$ be the current zero set. Assume that $s < t$; otherwise, there is no more informative collision. Let P be the linear polynomial corresponding to the new collision. Assume that $P \notin \mathsf{span}(\mathcal{Z})$. This implies that P is nonzero in the quotient ring $\mathbb{Z}_N[X_1, ..., X_t]/\mathsf{span}(\mathcal{Z}) \simeq \mathbb{Z}_N[L_1, ..., L_{t-s}]$ for some linear polynomials $L_1, ..., L_{t-s}$, and each variable L_i is uniform random over random choice of $x_1, ..., x_t$ conditioned on $Q_1, ..., Q_s = 0$, making P uniform over $\mathbb{Z}_N$. That is, $P = 0$ holds and is informative with probability $1/N$. □

For the readability, we restate the lower bound.

Theorem B.1. *Let $\mathcal{G}$ be a cyclic group of prime order. Let $\mathcal{A}_{m\text{-}\mathsf{MDL}}$ be an m-MDL algorithm in the GGM having at most T group operation gates. It holds that:*

$$\Pr_{\mathcal{A}_{m\text{-}\mathsf{MDL}}, \mathbf{x}}\left[\mathcal{A}^{\mathcal{G}}_{m\text{-}\mathsf{MDL}}(g, g^{\mathbf{x}}) \to \mathbf{x}\right] = O\left(\left(\frac{e(T+2m)^2}{2m|\mathcal{G}|}\right)^m\right).$$

Proof. As seen in the original proof of Theorem 3.4, we can assume that the algorithm finds m informative collisions to solve the m-MDL problem. Let T be the number of group operations.

By Lemma 3.2, each equality gate induces an informative collision with probability at most $1/N$. We further assume that the algorithm never applies the equality gates to the predictable inputs. This makes the probability that each equality gate is informative equal to $1/N$ independent from the previous equality gates.

Let E be the number of equality gates, which is at most $\binom{T+2m}{2} \le \frac{(T+2m)^2}{2}$. Assume that $\binom{T+2m}{2} \le mN$, otherwise the upper bound becomes larger than 1. Let C be the number of informative collisions during the algorithm and $\mu = \mathbb{E}[C] = \frac{E}{N}$. Let $\delta = \frac{mN}{E} - 1$. Note that $\delta\mu \le (1+\delta)\mu = m$. By the multiplicative Chernoff bound, we have

$$\begin{aligned}\Pr[C \ge m] &\le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu = \frac{e^{\mu\delta}}{(mN/E)^m} \\ &\le \frac{e^m}{(mN/E)^m} = \left(\frac{eE}{mN}\right)^m \\ &\le \left(\frac{eE}{mN}\right)^m \le \left(\frac{e(m+2T)^2}{2mN}\right)^m.\end{aligned}$$

Since this is an upper bound of the success probability of the m-MDL algorithm, it concludes the proof. □

C Equivalence between GGMs

This section proves that any single-stage problems secure in the Maurer-style (or type-safe) generic group model are also secure in the Shoup-style (or random representation) generic group model. The proof is essentially the same as [Zha22, Theorem 3.5] with some additional finer analysis.[7]

We call the generic algorithms described in Section 3 by type-safe (TS). We consider another style of generic algorithm that is called random representation (RR) introduced in [Sho97]. In this model, a set $S \in \{0,1\}^*$ (with the known maximal length of elements) is given public, and a random injection $L : \mathbb{Z}_N \to S$ is chosen, which is called by the *labeling function*. $L(x)$ is understood as a group element g^x. A generic algorithm in the random representation model is able to make the following queries:

Labeling Query. It takes $x \in \mathbb{Z}_N$ as input and outputs $L(x)$.

Group Operation Query. It takes $\ell_1, \ell_2 \in S$ and a single bit b as input. If there exist $x_1, x_2 \in \mathbb{Z}_N$ such that $L(x_1) = \ell_1$ and $L(x_2) = \ell_2$, it outputs $L(x_1 + bx_2)$. Otherwise, it outputs $\perp$.

We count the number of queries as a unit cost. A generic algorithm in this model is denoted by $\mathcal{A}^{\mathcal{G}_{RR}}$. Note that there is no equality query in this model, which can be done by comparing the labels without accessing the oracle. If an algorithm only makes queries with the inputs that it received before by some queries or input, then we call it *faithful*.

The following lemma shows that, when considering a single-stage game as in this paper, a faithful generic algorithm in the RR model is essentially the same as one in the TS model, but there is a subtle difference otherwise. We write $L(\mathbf{x}) = (L(x_1), ..., L(x_m))$ for $\mathbf{x} = (x_1, ..., x_m) \in \mathbb{Z}_N^m$.

Theorem C.1. *Let f be a function that takes an element in $\mathbb{Z}_N^m$ as input, $p > 0$, and D a distribution over $\mathbb{Z}_N^m$. Suppose for any generic algorithm $\mathcal{B}$ in the TS model with $T + \Lambda$ group operation complexity, it holds that*

$$\Pr_{\mathcal{B},\mathbf{x}\leftarrow D}[\mathcal{B}^{\mathcal{G}}(g^{\mathbf{x}}, \mathsf{aux}) = f(\mathbf{x})] \le p$$

where aux *is a bit string, Λ is to be specified and suppose that f includes k group element wires.*

[7] In the original paper, the author only considers the polynomially-bounded algorithms and negligible advantage. We need to consider more fine-grained equivalence for the exact advantage and any number of group operations.

Then, in the RR model, the following inequality holds

$$\Pr_{\mathcal{A},L,\mathbf{x}\leftarrow D}[\mathcal{A}^{\mathcal{G}_{RR}}(L(\mathbf{x}),\mathsf{aux}) = f(\mathbf{x})] \leq p - \Delta$$

where

- *for a faithful generic algorithm $\mathcal{A}$ with T queries and $\Lambda = \Delta = 0$, and*
- *in general, for a generic algorithm $\mathcal{A}$ with T queries such that at most t labels that are not given to $\mathcal{A}$ before, where $\Lambda = tr$ and $\Delta = r \cdot \left(\frac{T}{N}\right)^r$ for any positive integer r.*

In particular, when $T = N^{1-1/c}$ for some constant $c > 0$ and $p \geq 1/N$, we can choose $r = 2c$, which asserts that the asymptotic results equally hold.

Proof. Toward contradiction, we assume that there exists a generic algorithm $\mathcal{A}$ in the RR model with the winning probability larger than $p-\Delta$. We first consider the case that $\mathcal{A}$ is faithful. In this case, the algorithm $\mathcal{B}$ proceeds as follows. $\mathcal{B}$ initializes an empty table T, which will contain pairs (h, ℓ) for h in an element wire and $\ell \in S$. This will be interpreted as $L(x) = \ell$. We define the following subroutines of $\mathcal{B}$:

$\mathsf{FindLabel}(h)$**:** It takes an element wire containing h as input. It searches for a pair $(h', \ell) \in T$ with $h = h'$ using the equality gates. If such a pair exists, it returns ℓ. Otherwise, it samples a random $\ell \in S$ conditioned on ℓ not being in the table T. It adds (h, ℓ) to T and returns ℓ.

$\mathsf{FindElement}(\ell)$**:** It searches for a pair $(h, \ell') \in T$ with $\ell = \ell'$. If such a pair exists, it returns h on an element wire. Otherwise, it generates an element wire containing $\perp$ and adds $(\perp, \ell)$ to ℓ. It returns $\perp$. (This case does not occur for the faithful algorithms.)

$\mathcal{B}$ executes $\mathcal{A}$ and processes the queries from $\mathcal{A}$ and the inputs/outputs as follows.

- Given the problem instance, $\mathcal{B}$ parses it into a list L of element wires. For each element wire $h \in L$, $\mathcal{B}$ runs $\ell \leftarrow \mathsf{FindLabel}(h)$ and sends ℓ to $\mathcal{A}$ as a part of input corresponding to h.
- For a labeling query x from $\mathcal{A}$, $\mathcal{B}$ constructs an element wire containing g^x using a labeling gate. Then it runs $\ell \leftarrow \mathsf{FindLabel}(g^x)$ and returns ℓ to $\mathcal{A}$.
- For a group operation query (ℓ_1, ℓ_2, b), $\mathcal{B}$ runs $h_1 \leftarrow \mathsf{FindElement}(\ell_1), h_2 \leftarrow \mathsf{FindElement}(\ell_2)$, and computes $h = h_1 \cdot h_2^b$ using a group operation gate. Then it runs $\ell \leftarrow \mathsf{FindLabel}(h)$ and returns ℓ to $\mathcal{A}$.
- The final output of $\mathcal{B}$ is identical to that of $\mathcal{A}$. Precisely, if $\mathcal{A}$ outputs $(\ell_1, ..., \ell_k, \tau)$ for labels $\ell_1, ..., \ell_k$ and a string τ, $\mathcal{B}$ runs $h_i \leftarrow \mathsf{FindElement}(\ell_i)$ for $i \in [k]$ and outputs $(h_1, ..., h_k, \tau)$.

Note that each labeling query and group operation query incurs a single element gate, thus the group operation complexity of $\mathcal{B}$ is the same as one of $\mathcal{A}$. To prove that $\mathcal{B}$ wins with probability at least p, we consider the following sequence of hybrid experiments.

H_0**.** In this hybrid, $\mathcal{A}$ interacts with the group oracle $\mathcal{G}_{RR}$. $\mathcal{A}$ wins with probability at least p by the assumption.

H_1**.** This hybrid is the same as H_0 except that the random injection L is lazily sampled. This is possible because $\mathcal{A}$ is faithful. In the perspective of $\mathcal{A}$, this is identical to H_0, thus the winning probability is the same as H_0.

H_2**.** Here, $\mathcal{A}$ is a subroutine of $\mathcal{B}$. The view of $\mathcal{A}$ is identical to that of H_1, and the translation between two models is done inside of $\mathcal{B}$. The winning probability of $\mathcal{B}$ is equal to that of $\mathcal{A}$, which is the same as in H_1.

This completes the proof for the faithful $\mathcal{A}$.

We then consider the general case. In this case, $\mathcal{A}$ may ask queries with the labels it never received. To remedy this, we need to modify the subroutine FindElement, taking the probability that such a label is valid (i.e., an image of L) into account. The modified subroutine is as follows.

FindElement$'(\ell)$**:** It searches for a pair $(h, \ell') \in T$ with $\ell = \ell'$. If such a pair exists, it returns h on an element wire. Otherwise, let $m := |\{(h, \ell) \in T : h \neq \perp\}|$, and it does the following:

- With probability $1 - (N - m)/(|S| - |T|)$, it generates an element wire containing $\perp$ and adds $(\perp, \ell)$ to ℓ. It returns $\perp$.
- With probability $(N - m)/(|S| - |T|)$, it does the following procedures t times: It randomly samples $x \in \mathbb{Z}_N$ and construct the corresponding element wire containing g^x, and searches for $(h, \ell') \in T$ with $h = g^x$ using the equality gates. If such a pair does not exist, it adds (g^x, ℓ) to T, returns g^x on an element wire and halts. Otherwise, it discards g^x, and samples a fresh $x \in \mathbb{Z}_N$ and repeats.

A single iteration of the second case of FindElement$'$ terminates with probability at least $1-m/N \geq 1-T/N$, and takes one labeling gate.

In this case, the algorithm $\mathcal{B}$ in the TS model is defined with FindElement$'$ instead of FindElement. This change makes $\mathcal{B}$ find the corresponding element to the label that is not previously given. The success probability computation is almost identical, but incurs $(2T + k) \cdot \left(\frac{T}{N}\right)^r$ errors in the success probability regarding the failure of FindElement$'$.[8] If we carefully count the number t of the labels that are not given before, the number of gates becomes $q + tr$ and $\Delta = r \cdot \left(\frac{T}{N}\right)^r$.

□

C.1 Lower bounds in the Random Representation GGM

Let $L : \mathbb{Z}_N \to S$ be the labeling function, which will be lazily sampled. We prove the RR GGM variant of Theorem 4.1 in this section.

Note that the proof of Theorem C.1 for the faithful case works well for the unknown-order group case. In other words, it suffices to focus on the algorithm's behavior to look for a new label that was not given to the algorithm before.

Theorem C.2. *Let $\mathcal{A}_{\mathsf{ord}}$ be an order-finding algorithm over $\mathcal{D}^{(n)}_{\mathsf{prime}}$ in the random-representation GGM with the group operation complexity T. It holds that*

$$\Pr_{\mathcal{A}_{\mathsf{ord}}, N, L}\left[\mathcal{A}^{\mathcal{G}_N}_{\mathsf{ord}}() \to N\right] = O\left(\frac{T^3}{2^n}\right).$$

Proof sketch. Suppose that t labels to queries that are not given to $\mathcal{A}$ before and also not corresponding to $\perp$. We let them $\ell_1 = L(x_1), ..., \ell_t = L(x_t)$ and $\mathbf{x} = (x_1, ..., x_t)$. We must maintain the representations of the elements of $\mathcal{A}_{\mathsf{ord}}$ as a polynomial in $\mathbb{Z}[X_1, ..., X_t]$ where X_i corresponds to x_i. The definition of informative collisions is a pair of elements that have the same labels but as the polynomials different, and their difference is not included in the span of the previous informative collisions. Note that the algorithm must find at least one informative collision. Let us assume that it is represented by

$$P(x_1, ..., X_t) = a_1 X_1 + ... + a_t X_t + c = 0,$$

where we can assume that $|a_i|, |c| \leq 2^T N$ by the same reason to the original proof.

We make the encoding scheme for $(N, \mathbf{x})$.

[8] If we assume that $T \leq N^{1-1/c}$ for some constant $c > 0$, then repeating $r = 2c$ times ensures that the probability of failure is $1/N^2$ for each FindElement$'$.

$\mathsf{Encode}(N, \mathbf{x})$**:** It runs $\mathcal{A}_{\mathsf{ord}}^{\mathcal{G}_N}()$ and computes the first informative collision c. It additionally includes $\mathbf{x}$ as a part of the encoding. Note that the probability that $M = P(x_1, ..., x_t) = 0 \bmod p$ for another n-bit prime p is $1/p$, and $|P(x_1, ..., x_t)| \leq (t+1)2^T N^2$. Let ℓ be the index of N among the divisor of M.

$\mathsf{Decode}(c, \ell, \mathbf{x})$**:** If $c = \perp$, it outputs a random sample from $\mathcal{D}_{\mathsf{prime}}^{(n)}$. Otherwise, it recovers the first informative collision and plugs $\mathbf{x}$ to $X_1, ..., X_t$ to compute $M = P(x_1, ..., x_t)$, and outputs the ℓ-th prime factor N'.

The encoding size is $3 \log T - \log \log N + \log \binom{|\mathcal{G}|}{t} + O(1)$, which should be larger than $\log \frac{2^n}{n} + \log \binom{|\mathcal{G}|}{t} + \log \epsilon$. Rearranging this concludes the proof. □