On the Complexity of Cryptographic Groups and Generic Group Models

Cong Zhang* Keyu Ji* Taiyu Wang* Bingsheng Zhang*
Hong-Sheng Zhou† Xin Wang‡ Kui Ren*

September 18, 2024

Abstract

Ever since the seminal work of Diffie and Hellman, cryptographic (cyclic) groups have served as a fundamental building block for constructing cryptographic schemes and protocols. The security of these constructions can often be based on the hardness of (cyclic) group-based computational assumptions. Then, the generic group model (GGM) has been studied as an idealized model (Shoup, EuroCrypt 1997), which justifies the hardness of many (cyclic) group-based assumptions and shows the limits of some group-based cryptosystems. We stress that, the importance of the *length* of group encoding, either in a concrete group-based construction or assumption, or in the GGM, has not been studied.

In this work, we initiate a systematic study on the complexity of cryptographic groups and generic group models, varying in different lengths of group encodings, and demonstrate evidences that "the length matters". More concretely, we have the following results:

- We show that there is no black-box/relativizing reduction from the CDH-secure groups (i.e., over such groups, the computational Diffie-Hellman assumption holds) with shorter encodings, to the CDH-secure groups with longer encodings, within the same security parameter. More specifically, given any arbitrary longer CDH-secure group, it is impossible to generically shorten the group encoding and obtain a shorter CDH-secure group within the same group order.
- We show that there is a strict hierarchy of the GGMs with different lengths of encodings. That is, in the framework of indifferentiability, the shorter GGM is *strictly stronger* than the longer ones, even in the presence of computationally *bounded* adversaries.

*The State Key Laboratory of Blockchain and Data Security, Zhejiang University, and Hangzhou High-Tech Zone (Binjiang) Blockchain and Data Security Research Institute, {congresearch,jikeyu,taiyuwang,bingsheng,kuiren}@zju.edu.cn
†Virginia Commonwealth University, hszhou@vcu.edu
‡Digital Technologies, Ant Group, wx352699@antgroup.com

Contents

1 Introduction

Provable security and black-box reduction. Over the past few decades, *provable security* has become a cornerstone of modern cryptography. As a key technique in this field, reductions are used to justify the security of a scheme based on a cryptographic primitive. Essentially, given an adversary that allegedly breaks the scheme, one can transform it into an adversary that successfully attacks the underlying primitive. Our focus is largely on *black-box reductions*, where the primitive and adversary are treated as black boxes, interacting only through their input-output behavior without examining their internal structure.

In the field of group-based cryptography, introduced by Diffie and Hellman in their seminal work [DH76], reductions are built on the security of *cryptographic groups*. This foundation has driven the community to explore cryptographic groups from multiple perspectives.

From an efficiency perspective. In the literature, with few exceptions, group-based cryptosystems are often built on cryptographic groups in an abstract and black-box manner, which means the underlying groups can be instantiated by any concrete ones as long as the desired security properties are fulfilled. For instance, the well-known public key encryption (PKE) scheme, the ElGamal encryption [ElG85], is chosen-plaintext attack secure (IND-CPA) w.r.t. any concrete prime-order cyclic group in which the decisional Diffie-Hellman (DDH) assumption holds.

In practice, when aiming for better efficiency in cryptosystems, we typically choose concrete groups with shorter descriptions. Specifically, ElGamal encryption uses the prime-order subgroup of $\mathbb{Z}_p^*$ for a prime p, where each group element typically has a bit-length of 3072 for 128-bit security [Bar20]. Alternatively, elliptic curves are gaining popularity, with NIST SP 800-186 [CMR$^+$23] recommending curves like Curve25519, which achieves 128-bit security with a 255-bit prime modulus. With standard point compression, each Curve25519 group element can be encoded in 256 bits.

This highlights a critical yet subtle issue that has long been overlooked by the community. That is, *the bit-length of a group element is not explicitly taken into account* when the group is utilized in a black-box manner. Note, in real-world applications, groups with shorter descriptions are often preferred to minimize communication and computation overhead. Hereby, we ask the following questions: Does the length of the group description matter when using it in a black-box manner? Is it possible to construct a group with a shorter description generically from groups with longer descriptions? For notation simplicity, throughout this work, we will use *shorter groups* and *longer groups*, to denote "groups with shorter descriptions" and "groups with longer descriptions," respectively.

From a security perspective. Unfortunately, despite the advancement of modern cryptography, to the best of our knowledge, there is a fundamental limitation in provable security—the inability of establishing *unconditional hardness* with respect to a *concrete* group. In the past decades, researchers have made significant efforts to explore various ways to demonstrate the hardness of those group-based problems, and one approach is through the class of generic algorithms.

In essence, generic algorithms do not explore the specific encoding of group elements, but instead treat them in a generic manner. Studying this class of algorithms is highly motivated, since several well-known algorithms such as the baby-step/giant-step algorithm [PH78] and Pollard's rho algorithm [Pol78] fall within this classification. To formally describe generic algorithms, ever since the initial work by Nechaev [Nec94], variants of generic group models (GGMs) have been proposed. In Shoup's GGM [Sho97], the group is conceptualized as a random injective encoding from the additive group $\mathbb{Z}_N$ into bit strings uniformly sampled from a set S, where algorithms are allowed to retrieve group encodings and perform group operations, through oracle access. In Maurer's GGM [Mau05], the group is modeled as pointers with respect to a stateful register, where group encodings are the handles (or the indexes) of the register. Within both models, we can establish unconditional hardness, thereby validating the security of cryptographic groups.

When it comes to the study of the lengths of group encodings in the GGMs, Maurer, Portmann, and Zhu [MPZ20] initiate the models varying in the length of the group encoding, and illustrate a *partial hierarchy* of the GGMs, wherein any adversary within the GGM with a longer group encoding (below we denote it as "the longer GGM" for simplicity) can be converted into an adversary within the GGM with a shorter group encoding (below we denote it as "the shorter GGM" for simplicity). Despite the partial hierarchy, the connection and distinction between the longer GGM and the shorter GGM remains unexplored, which hinders a comprehensive interpretation and comparison of the numerous positive and negative results in the GGM.

To deepen our understanding on cryptographic groups, we ask the following question:

Will the longer group/GGM and the shorter group/GGM yield the same complexity?

1.1 Our Results

In this work, we initiate a fine-grained study of cryptographic groups and generic group models with different lengths of group encodings. Specifically, we give evidences that:

- There is a black-box separation between the shorter CDH-secure groups and the longer CDH-secure groups *with the same security parameter*; in other words, given longer CDH-secure groups, one cannot build a shorter CDH-secure group with the same group order from any standard techniques;
- The shorter GGMs are strictly stronger than the longer GGMs, even in the presence of *computationally bounded* adversaries.

To illustrate our findings, we first formalize the notions of groups/GGMs, parameterized by (N, m)[1], where N and m denote the order of the group and the length of the group encodings[2], respectively. More concretely, we respectively denote the (parameterized) groups and GGMs as $\mathcal{P}^{\mathsf{CDH}}_{N,m}$ and $\mathcal{G}_{N,m}$.

To establish the black-box separation between $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ and $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ where m_2 is much larger than m_1, we apply the common technique, namely, the relativizing separation. Concretely, we identify an idealized oracle $\mathcal{O}$ and prove that the longer CDH-secure groups exist relative to $\mathcal{O}$, but the shorter one does not exist. In our strategy, we set this oracle to be the GGM with longer group encodings, namely $\mathcal{G}_{N,m_2}$. At the first glance, this seems impossible, because the GGM is designed as the idealized model for cryptographic groups, and the GGM justifies the CDH by having the unconditional lower bound of the hardness. Fortunately, we observe that the analysis becomes subtle when the "length" is involved.

Theorem 1 (Main Theorem, informal)**.** *Consider $m_1 < m_2$. The shorter CDH-secure groups $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ are black-box separated from the longer CDH-secure groups $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$. Concretely,*

- *$\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist in the generic group model $\mathcal{G}_{N,m_2}$;*
- *$\mathcal{G}_{N,m_2}$ implies $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$.*

Remark 1. *Careful readers might wonder what is the relationship between the longer groups and the shorter groups in which* the discrete logarithm problem *is assumed to be hard. We emphasize that, due to technical challenges*[3]*, the relationship between the longer and shorter groups remains unknown—neither positively nor negatively established. We leave it as an open problem.*

Next, we turn our attention to understanding the relationship between the GGMs with different lengths of encoding. Based on the trivial observation that "$\mathcal{G}_{N,m_1}$ implies $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$", we immediately note that the shorter GGM, $\mathcal{G}_{N,m_1}$, and the longer GGM, $\mathcal{G}_{N,m_2}$ do not yield the same black-box complexity. However, when attempting to grasp the relationship between two idealized models, solely relying on black-box complexity might not provide us a comprehensive understanding. Essentially, the black-box complexity of a model only demonstrates the limit of standard-model cryptographic systems it implies and considers the computationally unbounded adversary.

[1] Typically N is sufficiently large, and $2^m \geq N$.

[2] By the length of group encoding, we mean that the binary length of the longest canonical representation for all group elements. For instance, let $\mathbb{G}$'s be a group such that the order is 3 and the canonical representation of the group elements is {00, 111, 0101}, then the length of $\mathbb{G}$, denoted as $\mathsf{len}_{\mathbb{G}}$, is 4.

[3] It is still unclear that whether discrete logarithm implies key agreement or not yet.

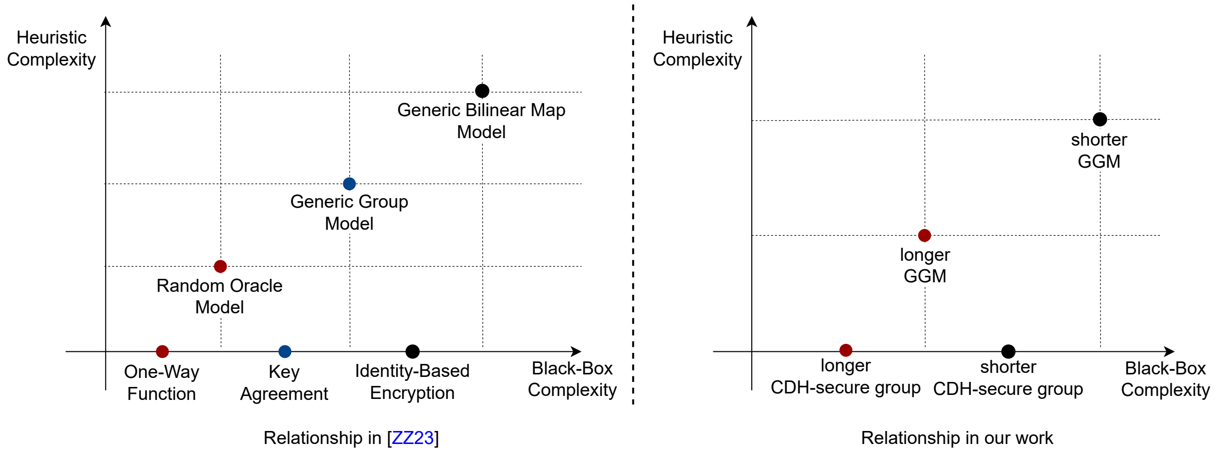

Figure 1: Relationship between idealized models.

To supplement this, Zhang and Zhandry [ZZ23] propose an orthogonal perspective to the black-box complexity, namely the heuristic complexity. It considers computationally bounded adversaries, thereby excluding the impact of all standard-model cryptosystems on the complexity. We investigate "the length matters" of GGMs within this new perspective, showing that:

Theorem 2 (Hierarchy of GGMs, informal)**.** *In the framework of indifferentiability, the GGM with shorter encodings is strictly stronger than the GGM with longer ones, even against computational bounded adversaries.*

To make it clearer, we show our results in Fig. 1. Following the notions in [ZZ23], we provide evidence that shorter GGMs statistically imply longer ones, whereas the existence of longer GGMs does not computationally imply the existence of shorter ones. More concretely, there exists an indifferentiable construction of a longer generic group with oracle access to shorter generic group unconditionally; whereas, as long as the difference in encoding lengths is sufficiently large, there does not exist an indifferentiable construction of a shorter generic group from a long generic group, even with additional computational assumptions.

1.2 Interpretation

Below, we offer interpretations of our findings.

From the perspective of black-box separation. Our results will bring the research community a better understanding of the cryptographic groups and the generic group models, from the perspective of the black-box reduction/separation[4]. In literature, generic group models have been frequently used to show the impossibility of constructing advanced group-based cryptosystems. Examples include identity-based encryption (IBE) [PRV12, SGS21, Zha22], indistinguishable obfuscation (iO) [MMN16], registration-based encryption (RBE) [HMQS23], accumulators [SGS20], order revealing encryption (ORE) [ZZ18], verifiable delay functions (VDF) [RSS20], and digital signature [DHH$^+$21]. Most of the separation results (e.g., [MMN16, ZZ18, RSS20, SGS20]) are established in Maurer's GGM. Meanwhile, Zhandry [Zha22] illustrates the limits of Maurer's GGM by proving that there are many natural group-based cryptographic schemes (e.g., efficient IND-CPA secure PKE) cannot be modeled by Maurer's GGM, and motivates the line of research, i.e., separations in Shoup's GGM (e.g., IBE in [Zha22] and RBE in [HMQS23]).

Our results demonstrate the *first* evidence that the generic group model can also be used to show the impossibility of constructing plain cryptographic groups, varying in distinct length of group encodings. Speaking of the "lengths" in cryptographic primitives, prior to our work, Garg et.al. [GMM17] prove that

[4]In this work, when talking about the black-box reduction/separation, we mean that the *fully* black-box reduction/separation that is explicitly defined in [RTV04].

there is no iO construction from the single-key functional encryption (FE), if the output length of the functions is much shorter than the length of the ciphertexts[5]. Therefore, we believe that, our result would motivate the community to study the "lengths" in fundamental primitives (e.g., PKE).

However, when delving deeper into our analysis, we stress that our separation results have a limitation. That is, we only establish the separations between the shorter CDH-secure groups and the longer CDH-secure groups under the condition that those groups yield the same security-parameter, which indicates that our separations are somehow security-parameter *dependent*.

For readability, we now explain the limitation through a concrete example. Let λ and λ' be two security parameters. Let p and p' be two primes where $\lfloor \log p \rfloor = \lambda$ and $\lfloor \log p' \rfloor = \lambda'$. Let $\mathbb{G}_1$ be a CDH-secure cryptographic group where the group order is p and the length is 2λ. Let $\mathbb{G}_2$ be another CDH-secure cryptographic group where the group order is p' and the length is $4\lambda'$. Apparently, $\mathbb{G}_1$ is the shorter group and $\mathbb{G}_2$ is the longer one. According to our findings, if $\lambda' \geq \lambda$, then one cannot generically build $\mathbb{G}_1$ from $\mathbb{G}_2$. Unfortunately, if $\lambda' < \lambda$ (say, $\lambda' = \frac{1}{3}\lambda$, indicating that $4\lambda' = \frac{4}{3}\lambda < 2\lambda$), then the relationship between $\mathbb{G}_1$ and $\mathbb{G}_2$ becomes unclear.

In contrast, most known separations are security-parameter *independent*. Take the separation of IBE in Shoup's GGM [Zha22] for instance; according to Zhandry's analysis, we have that for any sufficiently large λ and λ', one cannot generically build an IBE along with security-parameter λ', in Shoup's GGM with security-parameter λ. In order to establish a complete black-box separation (i.e., in the sense of security-parameter independent) between shorter groups and longer groups, novel techniques must be developed to resolve the limitation; we leave this as an important open problem.

Next, we justify that despite of the limitation, our results are interesting and important. First, when it comes to the problem that building a cryptographic group (say, $\mathbb{G}_1$) from another one (say, $\mathbb{G}_2$), it is natural to study the cases that: (1) $\mathbb{G}_1$ and $\mathbb{G}_2$ are with the same group order; (2) the order of $\mathbb{G}_1$ is a factor of $\mathbb{G}_2$[6]. Second, to the best of our knowledge, we are aware of no technique that can be used to generically build $\mathbb{G}_1$ from $\mathbb{G}_2$ if the group orders of $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct and co-prime. Therefore, we stress that our separations do capture the *natural* settings.

Moreover, our results serve as the first attempt to pin down the "lengths" problem for a fundamental primitive (i.e., cryptographic groups), which might open up new research directions (say, the "lengths" problem for other fundamental primitives). Below, for the ease of exposition, when we say the black-box separation between groups, we always mean the one with the same security parameter.

From a heuristic perspective. Our results will deepen our understanding of the generic group models from the perspective of heuristic complexity. Inspired by [MRH04, ZZ23], an idealized model can be interpreted through two orthogonal perspectives: the black-box complexity and the heuristic complexity, as depicted in Fig. 1.

For the heuristic aspect, initiated by Maurer et.al. [MRH04] and explicitly studied by Zhang and Zhandry [ZZ23], we consider the framework of indifferentiability against computationally bounded adversaries, where all cryptosystems that exist in the standard model are incorporated. Therefore, the perspective of heuristic is orthogonal to the one of black-box reduction/separation, and understanding the heuristic aspect of various idealized models is important for the relative security of cryptosystems based on idealized models. We establish a strict hierarchy of GGMs from this perspective and prove that the shorter GGM is strictly stronger than the longer one.

In the following, we will give an overview of our approach to comparing the various primitives/models, varying in different lengths of encodings, and our solutions for separating them.

1.3 Technical Overview

Separation between cryptographic groups. Given two cryptographic primitives $\mathcal{P}$ and $\mathcal{Q}$, the typical technique to establish the black-box separation is "relativizing separation" [IR89]. That is, we find a proper

[5]The separation is established under the condition that one-way functions (OWFs) exist and **NP** $\not\subseteq$ **coAM**.

[6]This case indicates that the security parameter of $\mathbb{G}_2$ is bigger than $\mathbb{G}_1$'s, and fortunately our analysis does capture such a case.

oracle $\mathcal{O}$ and prove that the primitive $\mathcal{P}$ exists relative to $\mathcal{O}$ but $\mathcal{Q}$ does not. In our setting, we consider the primitives $\mathcal{P}$ and $\mathcal{Q}$ to be the longer CDH-secure group and shorter one, respectively.

Compared to prior works, the main technical challenge is that, we need to show the gap between two primitives within the same security game (i.e., the CDH game), rather than within different games[7]. The first obstacle is to find a proper oracle. Apparently, the random oracle does not serve our purpose, because the random oracle is weak and there is no construction for CDH-secure groups in the random oracle model.

Our idea is to use a *stronger* oracle, the generic group model. At the first glance, this is impossible, because GGM implies CDH trivially! Fortunately, the GGMs varying in length of group encodings might also yield different levels of complexity, and thus we set this oracle to be the longer GGM within the same security parameter. Concretely, we denote the shorter groups, the longer groups and the longer GGM as $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}, \mathcal{P}^{\mathsf{CDH}}_{N,m_2}, \mathcal{G}_{N,m_2}$, respectively; recall that $m_2 > m_1$; and we prove that:

- $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ exists relative to $\mathcal{G}_{N,m_2}$;
- $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist relative to $\mathcal{G}_{N,m_2}$.

As the former statement is trivial, below we only explain the latter one. To show that $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist in $\mathcal{G}_{N,m_2}$, it suffices to construct an adversary $\mathcal{A}$ that breaks the CDH game for any construction of shorter group relative to $\mathcal{G}_{N,m_2}$. Due to technical difficulties, we switch to an alternative path. First we pin down a new primitive—non-interactive key exchange (NIKE) with shorter public key, denoted as $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$[8]. Then we prove that:

1. $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ implies $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$;
2. $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ does not exist relative to $\mathcal{G}_{N,m_2}$.

As the first statement is straightforward, we will prove the second one. Essentially, $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$, in and of itself, is a key agreement scheme. Next, we give a brief explanation of the separation between NIKE and the random oracle [BKSY11] and then demonstrate how to incorporate the ideas into our analysis. Let $\mathcal{H}$ be a random oracle and $\Pi^{\mathcal{H}} := (\mathsf{KGen}^{\mathcal{H}}, \mathsf{SHK}^{\mathcal{H}})$ be an NIKE scheme with *perfect* correctness. Assuming that the algorithms KGen and SHK make at most q queries, we then construct the adversary $\mathcal{A}$ as follows[9]. Let Alice and Bob be two honest parties. Given the transcript of an execution between Alice and Bob, i.e., pk_A and pk_B, in the present of $\mathcal{H}$, the adversary $\mathcal{A}$ maintains a set $S_{\mathsf{que\text{-}res}}$ of query/response pairs of $\mathcal{H}$, and a multiset S_{key} of candidate keys, both initialized to be $\emptyset$. The adversary $\mathcal{A}$ then runs $4q + 1$ iterations of the following attack:

- *Simulation Phase.* The adversary $\mathcal{A}$ searches a proper view of Alice that is consistent with pk_A and $S_{\mathsf{que\text{-}res}}$. Specifically, this view contains the randomness r^*_A used by KGen and SHK, as well as a set of oracle queries/responses $\hat{S}_{\mathcal{A}}$ made by KGen and SHK. The set $\hat{S}_{\mathcal{A}}$ is chosen to be consistent with $S_{\mathsf{que\text{-}res}}$, but it is unnecessary to be consistent with the true oracle $\mathcal{H}$. Let key be the value computed by $\mathsf{SHK}(r^*_A, \mathsf{pk}_B)$. Now, $\mathcal{A}$ adds key into S_{key}.
- *Update Phase.* The adversary $\mathcal{A}$ makes all queries in $\hat{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$ to the oracle $\mathcal{H}$, and adds the corresponding pairs into $S_{\mathsf{que\text{-}res}}$.

Finally, $\mathcal{A}$ outputs the majority of the keys in S_{key}. Next, we explain why $\mathcal{A}$ recovers the key. Let S_B denote the queries made by Bob in the real execution of the key exchange protocol. In a given iteration, there are two events:

- Event 1 (**Bad event**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$ s.t. $\mathsf{que}_A = \mathsf{que}_B$ but $\mathsf{res}_A \neq \mathsf{res}_B$.

[7]In [IR89], two different games, the one-wayness game and the key recovery attack game, are considered.
[8]Here, m_1 means the length of the public key; please find the formal definition in Section 2.1.
[9]The adversary here is computational unbounded but query-efficient.

- Event 2 (**Good event**): $\forall(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$, we have that if $\mathsf{que}_A = \mathsf{que}_B$, then $\mathsf{res}_A = \mathsf{res}_B$.

Note that event 1 only occurs in at most $2q$ iterations because $|S_B| \leq 2q$ and once it happens, the update phase would absorb at least one pair $(\mathsf{que}_B, \mathsf{res}_B) \in S_B$ into $S_{\mathsf{que\text{-}res}}$. For event 2, we observe that, when it occurs, there is another oracle $\tilde{\mathcal{H}}$ that is consistent with both $\hat{S}_{\mathcal{A}}$ and S_B. Based on the perfect correctness, we have that the shared key computed in that iteration is valid. Moreover, event 2 occurs in at least 2q+1 iterations, indicating that the majority in S_{key} is valid.

However, when it comes to the GGM, the attack fails. Comparing to ROM, there are two kinds of queries in GGM, namely the labeling query $(x, \mathcal{G}^{\mathsf{label}}_{N,m_2}(x))$ and the addition query $(\mathcal{G}^{\mathsf{label}}_{N,m_2}(x), \mathcal{G}^{\mathsf{label}}_{N,m_2}(y), \mathcal{G}^{\mathsf{label}}_{N,m_2}(x+y))$. Therefore, we should define S_B that covers all the group encodings that appear in the queries (both labeling and addition) with the discrete logarithms (Bob might not know the value). Then, in a given iteration, there are three events:

- Event 1 (**Bad event**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$ s.t. $\mathsf{que}_A = \mathsf{que}_B$ but $\mathsf{res}_A \neq \mathsf{res}_B$.
- Event 2 (**Bad event**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$ s.t. $\mathsf{que}_A \neq \mathsf{que}_B$ but $\mathsf{res}_A = \mathsf{res}_B$.
- Event 3 (**Good event**): $\forall(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$, we have that if $\mathsf{que}_A = \mathsf{que}_B$, then $\mathsf{res}_A = \mathsf{res}_B$.

Note that event 1 and event 3 can be handled similarly as above. However, the fatal problem is that event 2 might always happen. In other words, we cannot find a GGM that is consistent with both $\hat{S}_{\mathcal{A}}$ and S_B, indicating that the above attack fails immediately.

More specifically, we note that the reason why event 2 happens is that, given pk_A and pk_B, algorithms can obtain valid group encoding without making labeling query[10]. Moreover, if algorithms *cannot* obtain valid group encodings without making labeling queries, then the GGM can be simulated by a stateful oracle that only provides labeling queries, as the addition queries can be easily converted into labeling queries. Such an oracle is close to the random oracle model and thus our goal is to design a mechanism that prevent extracting valid group elements without knowing the corresponding discrete logarithms.

Here we introduce our *length tool*, intuitively, if the length of the public key is much shorter than the group encoding (say, the length gap is at least $\omega(\log \lambda)$), then the public key would not carry enough information to recover the group encodings. This also explains why we choose NIKE other than general key agreement (say, multi-round KA), because the adversary only obtains two public keys in the setting of NIKE.

Concretely, let Q_{sk_A} and Q_{sk_B} be the set of the query/response pairs (only labeling queries[11]) made when running $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_A)$ and $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_B)$, respectively. Let h be the valid group encoding that an algorithm outputs, by having pk_A and pk_B, we then consider the following four cases:

- Case 1: (**Independent**) $h \notin Q_{\mathsf{sk}_A} \cup Q_{\mathsf{sk}_B}$.
- Case 2: (**Frequent**) $h \in Q_{\mathsf{sk}_A} \cap Q_{\mathsf{sk}_B}$.
- Case 3: (**Dependent but hard to extract**) $h \in Q_{\mathsf{sk}_A} \setminus Q_{\mathsf{sk}_B}$.
- Case 4: (**Dependent but hard to extract**) $h \in Q_{\mathsf{sk}_B} \setminus Q_{\mathsf{sk}_A}$.

For case 1, h is independent of pk_A and pk_B. Due to the sparseness of the group encodings in $\mathcal{G}_{N,m_2}$, no algorithm can output h except for negligible probability.

[10] This in fact is natural in group-based cryptosystem, take the ElGamal encryption scheme [ElG85] for instance, the public key itself is a valid group element.

[11] We stress that, for the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}$, without loss of generality, it only makes labeling queries. Essentially, the group encodings of $\mathcal{G}_{N,m_2}$ are sparse, which means any algorithm with inputs that are independent of $\mathcal{G}_{N,m_2}$ cannot obtain a valid group encoding without making labeling query, indicating that any addition query can be absorbed by the corresponding labeling query.

For case 2, it is apparent that pk_A and pk_B together might carry enough information for h. Fortunately, with high probability h is a frequent query, therefore the discrete logarithm of h can be easily obtained by repeatedly running $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\cdot)$ on sufficiently many random inputs.

For case 3 (or case 4), note that h is independent of pk_B, which means that only pk_A carries the information of h. Note that the length of pk_A is m_1 but length of h is m_2. Moreover, h is uniformly distributed over the probability of GGM. Therefore, conditioned on that $m_2 - m_1$ is sufficiently large, no algorithm can extract such an h except for negligible probability.

The above sketch is not precise; please find low-level details, in Section 3.

Hierarchy of GGMs. To establish the hierarchy of the generic group models against computational bounded adversaries, we formalize our goal in the framework of indifferentiability. Specifically, we prove that the shorter GGM (denoted as $\mathcal{G}_{N,m_1}$) statistically implies the longer one (denoted as $\mathcal{G}_{N,m_2}$), but the longer GGM does not computationally imply the shorter one.

$\mathcal{G}_{N,m_1}$ statistically implies $\mathcal{G}_{N,m_2}$. We first explain how $\mathcal{G}_{N,m_1}$ implies $\mathcal{G}_{N,m_2}$ against computationally unbounded adversaries. Let $\mathcal{H}$ be a random oracle that maps $\{0,1\}^* \to \{0,1\}^{m_2-m_1}$; as the first attempt, it is natural to design the labeling function as:

$$L^{\mathcal{G}_{N,m_1},\mathcal{H}}(x) := \mathcal{G}^{\mathsf{label}}_{N,m_1}(x) || \mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x)).$$

However, there always exists an efficient distinguisher that breaks the indifferentiability w.r.t. the above scheme. Specifically, in the ideal world, the distinguisher uniformly samples $x \in \mathbb{Z}_N$, makes a labeling query with x, and obtains $\mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$. Let str and str' be the first m_1 bits and the last $m_2 - m_1$ bits of $\mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$, respectively. Then the distinguisher makes a query to the simulator with input str and checks whether the response matches str'. Note that, without knowing x, the simulator cannot answer this query properly except for a negligible probability. To prevent the attack above, we enhance the power of the simulator. We involve an additional oracle, the random permutation oracle $\mathcal{E}$, that permutes $\{0,1\}^{m_2}$ with its inverse[12] $\mathcal{E}^{-1}$, and design the labeling function as:

$$L^{\mathcal{G}_{N,m_1},\mathcal{H},\mathcal{E}}(x) := \mathcal{E}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x) || \mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x))).$$

Careful readers may wonder why it works. Note that both $\mathcal{E}$ and $\mathcal{E}^{-1}$ are under full control of the simulator, which means that the distinguisher is independent of the value $\mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x))$ without making queries to the simulator. This extra information gained from these queries is exactly what the simulator requires for the proof to go through. In fact, with the aid of $\mathcal{E}$, we can even simplify the construction by replacing $\mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x))$ with a fixed string, say $0 \cdots 0$, concretely:

$$L^{\mathcal{G}_{N,m_1},\mathcal{E}}(x) := \mathcal{E}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x) || \underbrace{0 \cdots 0}_{m_2 - m_1}).$$

The addition algorithm can be easily constructed by applying the inverse oracle $\mathcal{E}^{-1}$. While the additional oracle $\mathcal{E}$ and its inverse $\mathcal{E}^{-1}$ have protected against certain natural attacks, we need to argue indifferentiability against all possible attacks. To do so, in Section 4.1, we use a careful simulation strategy for $\mathcal{G}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1}, \mathcal{E}$, and $\mathcal{E}^{-1}$, and prove indifferentiability through a careful sequence of hybrids.

Remark 2. *Careful readers might note that the building blocks of construction above contain both the shorter GGM $\mathcal{G}_{N,m_1}$ and an additional independent randome oracle, rather than the shorter GGM solely. Although we have that GGM implies ROM statistically [ZZ23], it is unclear to us that how to build an indifferentiable GGM plus an independent ROM from a single GGM. Therefore, we stress that our hierarchy of the GGM is established with the aid of an additional independent random oracle.*

Moreover, this even motivates an interesting research question that whether one single GGM implies multiple independent GGMs, comparing to the fact that the random oracle does.

[12]According to [HKT11], the random oracle and random permutation oracle with inverse are equivalent, therefore we take $\mathcal{E}$ and $\mathcal{E}^{-1}$ for granted.

$\mathcal{G}_{N,m_2}$ **does not computationally imply** $\mathcal{G}_{N,m_1}$**.** Suppose we have a purported construction $\Pi^{\mathcal{G}_{N,m_2}} := (L^{\mathcal{G}_{N,m_2}}, A^{\mathcal{G}_{N,m_2}})$ of a shorter group from a longer GGM. How could we prove that $\Pi^{\mathcal{G}_{N,m_2}}$ can be differentiated from $\mathcal{G}_{N,m_1}$ by a computationally bounded distinguisher?

Following the strategy in [ZZ23], we should find some security property P that holds for $\mathcal{G}_{N,m_1}$ but fails for *any* $\Pi^{\mathcal{G}_{N,m_2}}$. As explained in [ZZ23], any standard model assumption cannot serve as the property, and thus, this property P is set to be a variant of discrete logarithm problem, called *discrete log identification* (DLI). Intuitively, DLI is defined as: given $h := L(x)$, construct a (probabilistic, efficient, and query-free) circuit C such that $C(x)$ accepts with a high probability, but $C(x')$ rejects with a overwhelming probability on all $x' \neq x$. Apparently, the DLI problem is easy on any standard-model group: for any y, set $C(y)$ to be 1 if and only if $L(y) = h$, where $L(y) := g^y$ is computed as part of the circuit[13]. To establish the separation between GGM and ROM, Zhandry and Zhang prove that the DLI problem is also easy on any group built within the random oracle model. Intuitively, they "compile out" the random oracle $\mathcal{H}$ and design an attacker that can easily construct an oracle-aided circuit $C^{\mathcal{H}}(\cdot)$, breaking the DLI problem by computing $L^{\mathcal{H}}(\cdot)$. The subtlety is to anticipate the oracle queries that C will make to the random oracle model and have the attacker make the corresponding queries for itself. Concretely, given input $L^{\mathcal{H}}(x)$, the attacker runs the addition algorithm $A^{\mathcal{H}}(L^{\mathcal{H}}(y), L^{\mathcal{H}}(x-y))$ and $L^{\mathcal{H}}(\cdot)$ on several random inputs, records all queries/responses that were made, and hardcodes the queries/responses into the C to obtain an oracle-free circuit, which C outputs.

Below, we outline our method for integrating the aforementioned technique into the analysis within the longer GGM. The difficulty is that, our goal seems to conflict with the results in [ZZ23], as they have proven that the DLI problem is hard with respect to the generic group model. To bypass the obstacle, we here leverage the *length tool* again.

Consider computing $L^{\mathcal{G}_{N,m_2}}(x)$ from x, which in turn makes queries to the longer GGM, $\mathcal{G}_{N,m_2}$. Let Q_x be the set of query/response pairs made during the procedure of computing $L^{\mathcal{G}_{N,m_2}}(x)$. Similarly as above, we assume that, without loss of generality, each query/response pair $(\mathsf{que}, \mathsf{res}) \in Q_x$ is a labeling query. Consider computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$ where y and z are random, conditioned on $y + z = x$. The output of this addition is $L^{\mathcal{G}_{N,m_2}}(y+z) = L^{\mathcal{G}_{N,m_2}}(x)$. For each query/response pair $(\mathsf{que}, \mathsf{res}) \in Q_x$, there are roughly four possible cases:

- Case 1: The label $L^{\mathcal{G}_{N,m_2}}(x)$ does *not* depend on the response res at all;
- Case 2: The label $L^{\mathcal{G}_{N,m_2}}(x)$ depends on the response res, but with a overwhelming probability over the choice of y and z, res does not appear when computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$;
- Case 3: The label $L^{\mathcal{G}_{N,m_2}}(x)$ depends on the response res, and with a non-negligible probability over the choice of y and z, $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$ makes a "labeling" query to $\mathcal{G}_{N,m_2}$ on input que;
- Case 4: The label $L^{\mathcal{G}_{N,m_2}}(x)$ depends on the response res, and with a non-negligible probability over the choice of y and z, an "addition" query $(\mathsf{que}_1, \mathsf{que}_2, \mathsf{res})$ occurs when computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$.

Now we explain our approach of building the oracle-free circuit C. We collect queries into a list, denoted as $S_{\mathsf{que\text{-}res}}$, and hardcode $S_{\mathsf{que\text{-}res}}$ into the circuit C to make sure that $C(x)$ will be able to reconstruct $L^{\mathcal{G}_{N,m_2}}(x)$ without making any query to the oracle at all:

- In case 1 (**Non-sensitive query**), same as in [ZZ23], since $L^{\mathcal{G}_{N,m_2}}(x)$ does not depend on res, when computing $L^{\mathcal{G}_{N,m_2}}(x)$ we can just replace the response with a random string without affecting the ultimate labeling. Therefore, for any query not in $S_{\mathsf{que\text{-}res}}$, we will have C respond with a uniformly random string.
- In case 2 (**Sensitive but frequent query**), same as in [ZZ23], since $L^{\mathcal{G}_{N,m_2}}(x)$ does depend on res, this query is a sensitive query for the ultimate labeling. In this case, it must be that $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$ be able to extract res from the inputs, i.e., $L^{\mathcal{G}_{N,m_2}}(y)$ and $L^{\mathcal{G}_{N,m_2}}(z)$, which indicates that, with a high probability, $(\mathsf{que}, \mathsf{res}) \in Q_x \cap (Q_y \cup Q_z)$. On the other hand, x, y and z are pairwise independent, which

[13]Note that for standard-model groups, $L(y)$ denotes the value g^y for the fixed generator g, and here y is the discrete logarithm of h with respect to g.

means that $Q_x \cap Q_y$ and $Q_x \cap Q_z$ *only* contains "frequent" queries. Therefore, this query/response pair can be collected by running $L^{\mathcal{G}_{N,m_2}}(\cdot)$ on sufficiently many random inputs.

- In case 3 (**Sensitive labeling query**), same as in [ZZ23], $S_{\mathsf{que\text{-}res}}$ collects all the labeling queries that occur when running $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$. We know that, with a non-negligible probability $(\mathsf{que}, \mathsf{res})$ will be amongst the queries in $S_{\mathsf{que\text{-}res}}$. By repeating several times, we have that $(\mathsf{que}, \mathsf{res}) \in S_{\mathsf{que\text{-}res}}$ with a high probability.

- In Case 4 (**Sensitive addition query**), different from [ZZ23], the addition query, i.e., $(\mathsf{que}_1, \mathsf{que}_2, \mathsf{res})$ occurs, where que_1 and que_2 are two valid group encodings of $\mathcal{G}_{N,m_2}$. Although res appears in this query, collecting this kind of query is not usually useful for our purpose. Specifically, when running $L^{\mathcal{G}_{N,m_2}}(x)$, the algorithm might make labeling queries on points $(x_1, \ldots, x_q)$, whereas $S_{\mathsf{que\text{-}res}}$ might only store query/response pairs in the form of addition, i.e., $(\mathcal{G}_{N,m_2}(y_i), \mathcal{G}_{N,m_2}(z_i), \mathcal{G}_{N,m_2}(y_i + z_i))$, without explicitly knowing either y_i or z_i. As a result, $C(x)$ might fail to reconstruct $L^{\mathcal{G}_{N,m_2}}(x)$: when running $C(x) := L^{S_{\mathsf{que\text{-}res}}}(x)$, although C knows that $\mathcal{G}_{N,m_2}(x_i)$ exists in the database $S_{\mathsf{que\text{-}res}}$, it does *not* know which tuple corresponds to the correct one.

  To resolve the problem, we need to transform this addition query into a labeling query. Observe that if the discrete logarithms of que_1 and que_2 are known, then the transformation is trivial. Exploring deeper, during the procedure of computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$, the algorithm $A^{\mathcal{G}_{N,m_2}}$ can only extract valid group encodings in $Q_y \cup Q_z$[14]. Moreover, we have that $L^{\mathcal{G}_{N,m_2}}(y)$ and $L^{\mathcal{G}_{N,m_2}}(z)$ are independent of the responses that are in $Q_z \setminus Q_y$ and $Q_y \setminus Q_z$, respectively.

  Now, we *leverage the length tool*. Concretely, from $A^{\mathcal{G}_{N,m_2}}$'s perspective, $L^{\mathcal{G}_{N,m_2}}(y)$ is the only string that carries information of the valid group encodings $\in Q_y \setminus Q_z$. If $m_2 - m_1$ is sufficiently large, say $m_2 - m_1 \geq \omega(\log \lambda)$, where λ is the security parameter, then it is impossible for $A^{\mathcal{G}_{N,m_2}}$ to extract a valid group encoding from $Q_y \setminus Q_z$ except for a negligible probability, indicating that the valid group encodings that $A^{\mathcal{G}_{N,m_2}}$ can extract are in $Q_y \cap Q_z$. Having that x, y and z are pairwise independent, we know that queries in $Q_y \cap Q_z$ are frequent with a high probability, which can be easily captured as in case 2.

Next, we consider the value of $C(x')$ for $x' \neq x$. If we are lucky and $S_{\mathsf{que\text{-}res}}$ contains all sensitive queries of $Q_{x'}$, then $C(x') = L^{\mathcal{G}_{N,m_2}}(x') \neq L^{\mathcal{G}_{N,m_2}}(x)$, indicating that $C(x')$ rejects as desired. Otherwise, if $S_{\mathsf{que\text{-}res}}$ does not contain all the sensitive queries of $Q_{x'}$, then $S_{\mathsf{que\text{-}res}}$ would respond to the query with random value, which means that $C(x')$ computes an invalid label for x'. As explained in [ZZ23], the random response would only serve to inject further randomness into the label, and the invalid label would be unequal to $L^{\mathcal{G}_{N,m_2}}(x)$ with a high probability. Combining the above together, we build an oracle-free circuit that *only* accepts the discrete logarithm x.

The above sketch is not precise; please find the low-level details in Section 4.

The hierarchy is tight. To complement our results of the hierarchy, we next show that if $m_2 - m_1$ is *small*, then $\mathcal{G}_{N,m_1}$ and $\mathcal{G}_{N,m_2}$ are equivalent under the indifferentiability framework. To explain our idea, we illustrate the simplest case, where $m_2 - m_1 = 1$[15]. Let Trunc be the function that chops off the last bit of the input, we build an indifferentiable group in $\mathcal{G}_{N,m_2}$ as follows:

$$L^{\mathcal{G}_{N,m_2}}(x) := \mathsf{Trunc}(\mathcal{G}^{\mathsf{label}}_{N,m_2}(x));$$

$$A^{\mathcal{G}_{N,m_2}}(\mathsf{str}_0, \mathsf{str}_1) := \begin{cases} \mathsf{Trunc}(\mathcal{G}^{\mathsf{add}}_{N,m_2}(\mathsf{str}_0 || b_0, \mathsf{str}_1 || b_1)), & \text{if } \mathsf{str}_0 || b_0 \text{ and } \mathsf{str}_1 || b_1 \text{ are valid;} \\ \perp & \text{otherwise.} \end{cases}$$

[14]Other group encodings in $\mathcal{G}_{N,m_2}$ are independent of $L^{\mathcal{G}_{N,m_2}}(y)$ and $L^{\mathcal{G}_{N,m_2}}(z)$.

[15]We also require that group encodings in $\mathcal{G}_{N,m_2}$ are sparse, say $m_2 - \log N \geq \omega(\log \lambda)$.

For clarity, if there exist $b_0, b_1 \in \{0,1\}$ such that both $\mathsf{str}_0||b_0$ and $\mathsf{str}_1||b_1$ are valid, then the addition algorithm outputs $\mathsf{Trunc}(\mathcal{G}^{\mathsf{add}}_{N,m_2}(\mathsf{str}_0||b_0, \mathsf{str}_1||b_1))$, otherwise it aborts. Based on the fact that the group encodings of $\mathcal{G}_{N,m_2}$ are sparse, we know that for any string str, the probability that both $\mathsf{str}||0$ and $\mathsf{str}||1$ are valid is negligible, which indicates that the addition algorithm is well defined. Moreover, we prove that the construction above is indifferentiable from $\mathcal{G}_{N,m_1}$ in Section 4.3.

Due to the composition of indifferentiability, our results can be easily extended to the case that $m_2 - m_1 \leq \Theta(\log \lambda)$, which completes the entire picture of the hierarchy asymptotically.

1.4 Organization

In Section 2, we present the necessary notations, concepts, and definitions. We establish a separation between two CDH-secure groups with sufficiently large encoding length difference in Section 3. We then establish a hierarchy among GGMs with different encoding lengths in Section 4.

2 Preliminaries

Notation. For a finite set S, we denote a random sample s from S according to the uniform distribution as $s \xleftarrow{\$} S$. We say a positive function $\mathsf{negl}(\cdot)$ is negligible, if for all positive polynomial $p(\cdot)$, there exists a constant $\lambda_0 > 0$ such that for all $\lambda > \lambda_0$, it holds that $\mathsf{negl}(\lambda) < 1/p(\lambda)$. We say a function $\rho(\cdot)$ is noticeable in λ, if the inverse $1/\rho(\lambda)$ is polynomial in λ. We write $y \xleftarrow{\$} \mathsf{Alg}(I)$ to denote variable y that is obtained by running a randomized algorithm Alg on input I (which may consist of a tuple $I := (I_1, ..., I_n)$). If Alg is deterministic, we write "$\leftarrow$" instead of "$\xleftarrow{\$}$". By $x||y$, we mean the concatenation of strings x and y.

Algorithms. Denote $\lambda \in \mathbb{N}$ as the security parameter. Here we use a non-uniform circuit to formalize the model of computation. An algorithm Alg is a collection of circuits $\{C_\lambda\}_{\lambda\in\mathbb{N}}$ with domain Dom_λ and range Ran_λ, respectively. When considering interactive algorithms $(\mathsf{Alg}_1, \ldots, \mathsf{Alg}_n)$, algorithms are treated as a sequence of circuits $C^{(1)}_\lambda, C^{(2)}_\lambda, \ldots$, where the domain of $C^{(i)}_\lambda$ is denoted as $\mathsf{Dom}^{(i)}_\lambda = \mathsf{stat}^{(i)}_\lambda \times \mathsf{input}^{(i-1)}_\lambda$, the range of $C^{(i)}_\lambda$ is denoted as $\mathsf{Ran}^{(i)}_\lambda = \mathsf{stat}^{(i+1)}_\lambda \times \mathsf{output}^{(i)}_\lambda$. Here, $\mathsf{stat}^{(i)}_\lambda$ ($\mathsf{input}^{(i)}_\lambda$, $\mathsf{output}^{(i)}_\lambda$) is the space of the state (inputs, outputs) that $C^{(i)}_\lambda$ sends to $C^{(i+1)}_\lambda$, respectively.

Games. A game is initiated by a probabilistic interactive algorithm C, called a challenger, and a predicate function $\mathsf{pf} : \{0,1\}^* \to [0,1]$. The challenger takes the security parameter as input and interacts with k communicating-restricted parties $(\mathsf{Alg}_1, \ldots, \mathsf{Alg}_k)$. We call $\mathcal{A} := (\mathsf{Alg}_1, \ldots, \mathsf{Alg}_k)$ the adversary. In the end of the game, the challenger C outputs a bit b; if $b = 1$ we say the adversary wins the game, otherwise we say the adversary loses. Let $\mathrm{Cl}(\mathcal{A})$ be a class of adversary. We say a game $(\mathsf{C}, \mathsf{pf})$ is hard with respect to $\mathrm{Cl}(\mathcal{A})$, if for any adversary $\mathcal{A} \in \mathrm{Cl}(\mathcal{A})$, we have $\Pr[\mathcal{A} \text{ wins}] \leq \mathsf{pf} + \mathsf{negl}(\lambda)$.

Cryptosystems. A cryptosystem Σ consists of a set of algorithms, which typically are non-interactive. Here, Σ is accessible via two interfaces $\Sigma.\mathrm{hon}$ and $\Sigma.\mathrm{adv}$, where $\Sigma.\mathrm{hon}$ provides an honest interface through which the system can be accessed by all parties in a black-box manner, and $\Sigma.\mathrm{adv}$ models the adversarial access to the inner working part of Σ.

2.1 Primitives, Idealized Models, and Reduction Notions

In this work, we treat CDH-secure groups as cryptographic primitives, and explore black-box reduction between them with different lengths. First of all, we recall the definition of primitive formalized by [RTV04].

2.1.1 Cryptographic Primitives

Definition 1 (Cryptographic Primitive [RTV04])**.** *A primitive $\mathcal{P}$ is a pair $\langle \mathcal{F}_\mathcal{P}, \mathcal{R}_\mathcal{P} \rangle$, where $\mathcal{F}_\mathcal{P}$ is a set of functions $f : \{0,1\}^* \mapsto \{0,1\}^*$, and $\mathcal{R}_\mathcal{P}$ is a relation over pairs $\langle f, \mathcal{A} \rangle$ of a function $f \in \mathcal{F}_\mathcal{P}$ and an adversarial machine $\mathcal{A}$. (The set $\mathcal{F}_\mathcal{P}$ is required to contain at least one function which is computable by a* PPT *machine.)*

- Efficient implementation. *We say a function f* **implements** *$\mathcal{P}$ or is an implementation of $\mathcal{P}$ if $f \in \mathcal{F}_\mathcal{P}$. An* **efficient implementation** *of $\mathcal{P}$ is an implementation of $\mathcal{P}$ which is polynomial-time computable.*
- Secure implementation. *We say an adversarial machine $\mathcal{A}$* **$\mathcal{P}$-breaks** *$f \in \mathcal{F}_\mathcal{P}$ if $\langle f, \mathcal{A} \rangle \in \mathcal{R}_\mathcal{P}$. A* **secure implementation** *of $\mathcal{P}$ is an implementation of $\mathcal{P}$ such that no* PPT *adversarial machine $\mathcal{P}$-breaks f.*

We say the **primitive $\mathcal{P}$ exists** *if there is an efficient and secure implementation of $\mathcal{P}$.*

As mentioned before, we treat CDH-secure groups as a cryptographic primitive. Now we formalize this primitive by using the terms in [RTV04].

Definition 2 (CDH-Secure Groups)**.** *A CDH-secure group $\mathcal{P}^{\mathsf{CDH}}$ consists of the following pair $\langle \mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}, \mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}} \rangle$:*

1. *The set $\mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$ for specifying syntax and capturing the correctness property.*

   Here, the set $\mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$ consists of functions f, where f represents the group generation function *for generating group description of finite cycle groups. Concretely, we write $(\mathbb{G}, g, N, m) \overset{\$}{\leftarrow} f(1^\lambda)$, where $\mathbb{G}$ is a cyclic group of prime order N, g is a generator $\mathbb{G}$, and m is the length of group encoding (that is, each group element in $\mathbb{G}$ can be represented as an m-bit string).*

   We note that the correctness is guaranteed by the basic properties of the cyclic group.

2. *The relation $\mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}}$ for capturing the security property.*

   For function $f \in \mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$ and PPT *(adversarial) machine $\mathcal{A}$, we define $\langle f, \mathcal{A} \rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}}$ if there exists a polynomial $p(\cdot)$ such that $\Pr[\mathcal{A}(\mathbb{G}, g, N, m, h_1, h_2) = g^{x_1 x_2}] > 1/p(\lambda)$ for infinitely many λ.*

   Here, $(\mathbb{G}, g, N, m) \overset{\$}{\leftarrow} f(1^\lambda)$, and $h_1, h_2 \in \mathbb{G}$ are two uniformly chosen group elements where $h_1 = g^{x_1}$, $h_2 = g^{x_2}$, and $x_1, x_2 \in \mathbb{Z}_N$.

We say CDH-secure group $\mathcal{P}^{\mathsf{CDH}}$ exists, if there exists a function $f \in \mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$, it holds that no PPT *adversarial machine $\mathcal{A}$ such that $\langle f, \mathcal{A} \rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}}$. Often, we make the parameters, the order N and the encoding length m, explicit, and denote the CDH-secure group as $\mathcal{P}^{\mathsf{CDH}}_{N,m}$.*

Non-interactive key exchange (NIKE) was initially studied by Diffie and Hellman in their breakthrough paper [DH76]. We now describe this primitive by using the terms in [RTV04].

Definition 3 (Non-Interactive Key Exchange)**.** *A non-interactive key exchange protocol $\mathcal{P}^{\mathsf{NIKE}}$ consists of the following pair $\langle \mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}, \mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}} \rangle$:*

1. *The set $\mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$ for specifying syntax and capturing the correctness property.*

   Here, the set $\mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$ consists of functions f, where $f := (\mathsf{KGen}, \mathsf{SHK})$ represents

   - *the* public-key message function $\mathsf{KGen} : \mathcal{SK} \mapsto \mathcal{PK}$ *for generating the public-key message based on a randomly chosen private-key, where $\mathcal{PK}$ and $\mathcal{SK}$ are public-key space and private-key space, respectively.*
   - *the* shared key generation function $\mathsf{SHK} : \mathcal{PK} \times \mathcal{SK} \mapsto \mathcal{K} \cup \{\perp\}$ *for generating the shared key, where $\mathcal{K}$ is shared-key space, and $\perp$ denotes that the computation fails.*

   Concretely, for randomly chosen $\mathsf{sk} \overset{\$}{\leftarrow} \mathcal{SK}$, we write $\mathsf{pk} \leftarrow \mathsf{KGen}(\mathsf{sk})$, where pk is called public key. Furthermore, for randomly chosen $\mathsf{sk}' \overset{\$}{\leftarrow} \mathcal{SK}$, compute $\mathsf{pk}' \leftarrow \mathsf{KGen}(\mathsf{sk}')$. We write $\mathsf{shk} \leftarrow \mathsf{SHK}(\mathsf{pk}', \mathsf{sk})$ and $\mathsf{shk}' \leftarrow \mathsf{SHK}(\mathsf{pk}, \mathsf{sk}')$. Note that, when the shared key generation function fails, we write $\mathsf{shk} = \perp$ or $\mathsf{shk}' = \perp$.

We say correctness *is achieved if there exists an* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathsf{shk} \neq \perp \wedge \mathsf{shk}' \neq \perp \wedge \mathsf{shk} \neq \mathsf{shk}'\right] \leq \mathsf{negl}(\lambda)$$

When $\mathsf{negl}(\lambda) = 0$*, then we say* perfect correctness *is achieved.*

2. *The relation* $\mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}}$ *for capturing the security property against key-recovery attack (KRA).*

   For function $f := (\mathsf{KGen}, \mathsf{SHK}) \in \mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$ *and a* PPT *(adversarial) machine* $\mathcal{A}$*, we define* $\langle f, \mathcal{A}\rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}}$ *if there exists a polynomial* $p(\cdot)$ *such that* $\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{pk}') = \mathsf{SHK}(\mathsf{pk}', \mathsf{sk}) = \mathsf{SHK}(\mathsf{pk}, \mathsf{sk}') \neq \perp] > 1/p(\lambda)$ *for infinitely many* λ*.*

   Here, for randomly chosen $\mathsf{sk} \overset{\$}{\leftarrow} \mathcal{SK}$ *and* $\mathsf{sk}' \overset{\$}{\leftarrow} \mathcal{SK}$*, compute* $\mathsf{pk} \leftarrow \mathsf{KGen}(\mathsf{sk})$ *and* $\mathsf{pk}' \leftarrow \mathsf{KGen}(\mathsf{sk}')$*, respectively.*

We say non-interactive key exchange protocol $\mathcal{P}^{\mathsf{NIKE}}$ *exists, if there exists a function* $f \in \mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$*, it holds that no* PPT *adversarial machine* $\mathcal{A}$ *such that* $\langle f, \mathcal{A}\rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}}$*. When* $\mathcal{SK} = \mathbb{Z}_N$ *and* $\mathcal{PK} = \mathcal{K} = \{0,1\}^m$*, we make the parameters* N *and* m *explicit and denote the non-interactive key exchange protocol as* $\mathcal{P}^{\mathsf{NIKE}}_{N,m}$*.*

2.1.2 Idealized Models

In this subsection, we introduce idealized models including the Random Oracle Model (ROM) [BR93], the Random Permutation Model (RPM) [RS08], and the Generic Group Model (GGM) [Sho97]. In each idealized model, all entities including the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$, are provided with the access to the corresponding oracle. Below we will specify the behavior of the oracle in each idealized model.

Definition 4 (Random Oracle Model [BR93])**.** *Let* $\mathcal{I}_{*,S}$ *denote the set of functions* $h : \{0,1\}^* \to S$*, where* $S := \{0,1\}^n$ *for some integer* n*. The random oracle model* $\mathcal{H}$ *is an idealized model, sampling a random function* h *from* $\mathcal{I}_{*,S}$*. Every algorithm can query* x*, obtaining the corresponding value* $h(x) \in S$*.*

Definition 5 (Random Permutation Model [RS08])**.** *Let* $\mathcal{I}_{S,S}$ *denote the set of permutations* $\pi : S \to S$*, where* $S := \{0,1\}^n$ *for some integer* n*. The random permutation model* $\mathcal{E}$ *is an idealized model, sampling a random permutation* π *from* $\mathcal{I}_{S,S}$*. Every algorithm can query* $x \in S$ *with* $\mathcal{E}$ *for both* π *and its inverse* π^{-1}*, obtaining the corresponding value* $\pi(x) \in S$ *or* $\pi^{-1}(x) \in S$*.*

Definition 6 (Generic Group Model [Sho97])**.** *Denote by* $\mathcal{I}_{\mathbb{Z}_N,S}$ *the set of injections* $\sigma : \mathbb{Z}_N \mapsto S$*, where* $S := \{0,1\}^m$*. The generic group model* $\mathcal{G}_{N,m}$ *is an idealized model, sampling a random injection* σ *from* $\mathcal{I}_{\mathbb{Z}_N,S}$*, with functions* $\mathcal{G}^{\mathsf{label}}_{N,m}$ *and* $\mathcal{G}^{\mathsf{add}}_{N,m}$*. Concretely, for each query* $x \in \mathbb{Z}_N$*, the "labeling" function* $\mathcal{G}^{\mathsf{label}}_{N,m}$ *responds with a value* $\sigma(x) \in S$*. For a query* (g_1, g_2)*, the "adding" function* $\mathcal{G}^{\mathsf{add}}_{N,m}$ *answers as follows: if* $g_1 = \sigma(x_1)$ *and* $g_2 = \sigma(x_2)$ *for some* $x_1, x_2 \in \mathbb{Z}_N$*, replying by* $\sigma(x_1 + x_2)$*, and replying by* $\perp$ *otherwise.*

2.1.3 Notions of Reductions

To establish separations between primitives, in this paper, we follow two notions, *fully black-box reduction* and *relativizing reduction*, as formalized by Reingold, Trevisan, and Vadhan [RTV04].

Definition 7 (Fully Black-Box Reduction [RTV04])**.** *There exists a fully black-box reduction from a primitive* $\mathcal{P} := \langle \mathcal{F}_{\mathcal{P}}, \mathcal{R}_{\mathcal{P}}\rangle$ *to a primitive* $\mathcal{Q} := \langle \mathcal{F}_{\mathcal{Q}}, \mathcal{R}_{\mathcal{Q}}\rangle$*, if there exist* PPT *oracle machines* Π *and* $\mathcal{B}$ *such that:*

Correctness *For every implementation* $f \in \mathcal{F}_{\mathcal{Q}}$ *we have that* $\Pi^f \in \mathcal{F}_{\mathcal{P}}$*.*

Security *For every implementation* $f \in \mathcal{F}_{\mathcal{Q}}$*, if there exists a* PPT *oracle machine* $\mathcal{A}$ *such that* $\mathcal{A}^f$ $\mathcal{P}$*-breaks* Π^f*, then there exists a* PPT *oracle machine* $\mathcal{B}$ *such that* $\mathcal{B}^f$ $\mathcal{Q}$*-breaks* f*.*

In literature, a typical technique for black-box separation, say for primitives $\mathcal{P}$ and $\mathcal{Q}$, is relativizing separation, which means that there is no relativizing reduction between $\mathcal{P}$ and $\mathcal{Q}$. Reingold et al. [RTV04] indicate that fully black-box reduction implies relativizing reduction, referring to that the relativizing separation from $\mathcal{P}$ to $\mathcal{Q}$ indicates the corresponding fully black-box separation.

Definition 8 (Relativizing Reduction [RTV04]). *There exists a relativizing reduction from a primitive $\mathcal{P} := \langle \mathcal{F}_{\mathcal{P}}, \mathcal{R}_{\mathcal{P}} \rangle$ to a primitive $\mathcal{Q} := \langle \mathcal{F}_{\mathcal{Q}}, \mathcal{R}_{\mathcal{Q}} \rangle$, if for every oracle $\mathcal{O}$, the primitive $\mathcal{P}$ exists relative to $\mathcal{O}$ whenever $\mathcal{Q}$ exists relative to $\mathcal{O}$. A primitive $\mathcal{P}$ is said to exist relative to $\mathcal{O}$, if there exists $f \in \mathcal{F}_{\mathcal{P}}$ which has an efficient implementation when having access to the oracle $\mathcal{O}$ such that no* PPT *oracle machine with access to $\mathcal{O}$, can $\mathcal{P}$-break f.*

2.2 Indifferentiability

The framework of indifferentiability is proposed by Maurer, Renner, and Holenstein [MRH04], which formalizes a set of necessary and sufficient conditions for securely replacing one cryptosystem with another in an arbitrary environment. This framework is used to justify the structural soundness of various cryptographic primitives, including hash functions [CDMP05, DRS09], block ciphers [ABD+13, CHK+16, DSSL16, GWL23], domain extenders [CDMS10], authenticated encryption with associated data [BF18], and public key cryptosystems [ZZ20]. It can also be used to study the relationship between idealized models [ZZ23]. Within the context of the indifferentiability framework, it is customary to consider that a cryptosystem either implements certain ideal objects denoted as $\mathcal{F}$, or it is a construction denoted as $C^{\mathcal{F}'}$ that relies on underlying ideal objects $\mathcal{F}'$.

Definition 9 (Indifferentiability [MRH04]). *Let Σ_1 and Σ_2 be two cryptosystems and $\mathcal{S}$ be a simulator. The indifferentiability advantage of a distinguisher $\mathcal{D}$ against (Σ_1, Σ_2) with respect to $\mathcal{S}$ is*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}(1^\lambda) := \Pr[\mathrm{Real}_{\Sigma_1,\mathcal{D}}] - \Pr[\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}],$$

where games $\mathrm{Real}_{\Sigma_1,\mathcal{D}}$ and $\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}$ are defined in Fig. 2. We say Σ_1 is indifferentiable from Σ_2, if there exists an efficient simulator $\mathcal{S}$ such that for any efficient distinguisher $\mathcal{D}$, the advantage above is negligible. Moreover, we say Σ_1 is statistically indifferentiable from Σ_2, if there exists an efficient simulator such that, for any unbounded distinguisher $\mathcal{D}$, the advantage above is negligible.

```
Indifferentiability

Real_{Σ1,D}:                      HonestR(X)               |  Ideal_{Σ2,S,D}:                  HonestI(X)
b ← D^{HonestR,AdvR}              Return Σ1.hon(X).        |  b ← D^{HonestI,AdvI}             Return Σ2.hon(X).
Return b.                         AdvR(X)                  |  Return b.                        AdvI(X)
                                  Return Σ1.adv(X).        |                                   Return S^{Σ2.adv(·)}(X).
```

Figure 2: Indifferentiability of Σ_1 and Σ_2, where $\mathcal{S}$ is the simulator and $\mathcal{D}$ is the adversary.

Below, we also use the notations in [BF18] and consider the definition above to two systems with interfaces as:

$$(\Sigma_1.\mathrm{hon}(X), \Sigma_1.\mathrm{adv}(x)) := (\Pi^{\mathcal{F}_1}(X), \mathcal{F}_1(x)),$$
$$(\Sigma_2.\mathrm{hon}(X), \Sigma_2.\mathrm{adv}(x)) := (\mathcal{F}_2(X), \mathcal{F}_2(x)),$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ are two ideal objects sampled from their distributions and $\Pi^{\mathcal{F}_1}$ is a construction of $\mathcal{F}_2$ by calling $\mathcal{F}_1$. Maurer, Renner, and Holenstein prove the composition theorem for the framework of indifferentiability; for simplicity, we give a game-based formalization from [RSS11].

Theorem 3 (Composition Theorem [MRH04]). *Let $\Sigma_1 := (\Pi^{\mathcal{F}_1}, \mathcal{F}_1)$ and $\Sigma_2 := (\mathcal{F}_2, \mathcal{F}_2)$ be two systems that Σ_1 is indifferentiable from Σ_2 with respect to a simulator $\mathcal{S}$, then Σ_1 is as secure as Σ_2 for any single-stage game. More concretely, let* Game *be a single-stage game, then for any adversary $\mathcal{A}$, there is an adversary $\mathcal{B}$ and a distinguisher $\mathcal{D}$ such that*

$$\Pr[\mathsf{Game}_{\Pi^{\mathcal{F}_1},\mathcal{A}^{\mathcal{F}_1}}] \leq \Pr[\mathsf{Game}_{\mathcal{F}_2,\mathcal{B}^{\mathcal{F}_2}}] + \mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}.$$

The proof of Theorem 3 is straightforward; due to space limit, we skip it here. Next, we give the formal definition of the separation between two idealized models in the framework of indifferentiability against computational adversaries.

Definition 10 (Computational Indifferentiable Separation [MRH04, ZZ23])**.** *Let Σ_1, Σ_2 be two idealized models, we say Σ_2 is computationally indifferentiably separated from Σ_1 if for any efficient algorithm Π and any efficient simulator $\mathcal{S}$, there exists an efficient distinguisher $\mathcal{D}_{\Pi,\mathcal{S}}$ and a noticeable function ρ such that*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi^{\Sigma_1},\Sigma_2,\mathcal{S},\mathcal{D}_{\Pi,\mathcal{S}}}(1^\lambda) := \left|\Pr[\mathrm{Real}_{\Sigma_1,\mathcal{D}_{\Pi,\mathcal{S}}}] - \Pr[\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}_{\Pi,\mathcal{S}}}]\right| \geq \rho(\lambda).$$

Observe that, if an idealized model Σ_2 is computationally indifferentiably separated from another idealized model Σ_1, it means that, we cannot build a scheme Π^{Σ_1} such that Π^{Σ_1} is indifferentiable from Σ_2, even under arbitrarily strong computational assumptions.

3 Separation between Cryptographic Groups

In this section, we establish the separation between two CDH-secure groups, $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ and $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$, under the condition that both N and $(m_2 - m_1)$ are sufficiently large within the same security parameter.

Theorem 4 (Main Theorem)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter. Let N, m_1, m_2 be integers such that $N \geq 2^{\omega(\log\lambda)}$, $m_1 > \log N$ and $m_2 - m_1 \geq \omega(\log\lambda)$. Then there is no black-box reduction from $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ to $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$.*

Proof. To establish the theorem, we apply the so-called two-oracle technique [HR04]. Let PSPACE be a PSPACE-complete oracle. Essentially, we set $\mathcal{O} := (\mathsf{PSPACE}, \mathcal{G}_{N,m_2})$ and prove the following:

1. $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ exists relative to $\mathcal{O}$;
2. $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist relative to $\mathcal{O}$.

The former statement holds trivially as $\mathcal{G}_{N,m_2}$ implies $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ in the canonical manner. Therefore, it suffices to prove the latter one.

Lemma 1. *$\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist relative to $\mathcal{O}$.*

To establish the proof, we first pin down an intermediary primitive, i.e., $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ (within the same security parameter), defined in Section 2.1, and then prove that:

1. $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ implies $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$;
2. $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ does not exist relative to $\mathcal{O}$.

The first statement holds straightforwardly. Next, we establish our theorem by proving the following lemma.

Lemma 2. *$\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ does not exist relative to $\mathcal{O}$.*

Intuitively, to prove that $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ does not exist relative to $\mathcal{O}$, it is sufficient to build a PPT oracle adversary $\mathcal{A}^{\mathcal{O}}$ that breaks any construction $\Pi^{\mathcal{O}} := (\mathsf{KGen}^{\mathcal{O}}, \mathsf{SHK}^{\mathcal{O}})$. Observe that $\mathcal{A}^{\mathcal{O}}$ has access to a PSPACE-complete oracle, which means that $\mathcal{A}^{\mathcal{O}}$ implies a computationally unbounded but query-efficient adversary that only has access to $\mathcal{G}_{N,m_2}$[16]. Therefore, it suffices to construct such an adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}$. In Fig. 3, we illustrate the description of the adversary.

We first clarify some undefined notions: Let n be a sufficiently large integer that will be specified below. By $\left\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\right\} \xleftarrow{\mathsf{query}} \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(r_i)$, we mean that when running the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(r_i)$, the algorithm makes queries $(\mathsf{que}_1, \ldots, \mathsf{que}_q)$ to the oracle $\mathcal{G}_{N,m_2}$ and obtains $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$[17].

[16] Any computationally unbounded but query-efficient adversary can be simulated by a PPT oracle machine with access to a PSPACE-complete oracle, that is because what we need are specific labeling query-response tuples of GGM. These tuples can be picked by using a PSPACE-complete oracle. See [MM11] for more details.

[17] As explained above, we stress that $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}$ only makes labeling queries.

```
Adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}(\mathsf{pk}_A, \mathsf{pk}_B)$

$\mathcal{A}^{\mathcal{G}_{N,m_2}}(\mathsf{pk}_A, \mathsf{pk}_B)$:
$S_{\mathsf{key}} \leftarrow \emptyset$; $S_{\text{que-res}} \leftarrow \emptyset$; $r_1, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_N$;
Initial phase: //collecting frequent queries
    for $i = 1$ to $n$:
        $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\text{query}} \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(r_i)$
        $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}$;
for $i = 1$ to $12q + 1$: //running $12q + 1$ iterations
    Simulation phase: //searching a proper view
        Search a secret key $\widetilde{\mathsf{sk}}_A$ and a set of query-response tuples $\hat{S}_{\mathcal{A}}$ satisfying
        the following properties:
          Property 1: $\hat{S}_{\mathcal{A}}$ is consistent with $S_{\text{que-res}}$;
          Property 2: $\mathsf{pk}_A = \mathsf{KGen}^{S_{\text{que-res}} \cup \hat{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}_A)$;
          Property 3: $\hat{S}_{\mathcal{A}}$ only collects tuples of labeling queries and $|\hat{S}_{\mathcal{A}}| \leq 12q$;
          Property 4: $\hat{S}_{\mathcal{A}}$ is sufficient for SHK. That is, when running $\widetilde{\mathsf{shk}}_i \leftarrow \mathsf{SHK}^{S_{\text{que-res}} \cup \hat{S}_{\mathcal{A}}}(\mathsf{pk}_B, \widetilde{\mathsf{sk}}_A)$: (1) the
          set $S_{\text{que-res}} \cup \hat{S}_{\mathcal{A}}$ covers all labeling queries; (2) the set $S_{\text{que-res}} \cup \hat{S}_{\mathcal{A}}$ is able to convert any addition query
          into a labeling query properly. Let $\mathsf{que} = (h_1, h_2)$ be an addition query when running the shared-
          key algorithm, if either $h_1$ or $h_2$ is not covered in $\hat{S}_{\mathcal{A}} \cup S_{\text{que-res}}$, then responds with $\perp$; otherwise the
          set $S_{\text{que-res}} \cup \hat{S}_{\mathcal{A}}$ must cover the following three tuples: $(x_1, h_1), (x_2, h_2)$ and $(x_1 + x_2, h_3)$, and responds
          with $h_3$.
        $S_{\mathsf{key}} \leftarrow S_{\mathsf{key}} \cup \{\widetilde{\mathsf{shk}}_i\}$;
    Update phase: //updating the guessing labeling queries with valid encodings
        for each $(\mathsf{que}, \mathsf{res}) \in \hat{S}_{\mathcal{A}} \setminus S_{\text{que-res}}$: $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup (\mathsf{que}, \mathcal{G}^{\mathsf{label}}_{N,m_2}(\mathsf{que}))$;
Final phase: //outputting the guessing shared key
    return the majority value in $S_{\mathsf{key}}$.
```

Figure 3: The description of the adversary that breaks $\Pi^{\mathcal{G}_{N,m_2}}$.

Next, we prove that $\mathcal{A}^{\mathcal{G}_{N,m_2}}$ outputs the valid shared key with noticeable probability. Let $S_{B\text{-label}}$ be the set of the valid group elements that appear when running $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_B)$ and $\mathsf{SHK}^{\mathcal{G}_{N,m_2}}(\mathsf{pk}_A, \mathsf{sk}_B)$; those group elements are either the responses of labeling/addition queries or the valid inputs of the addition queries. It is apparent that $|S_{B\text{-label}}| \leq 6q$, due to the fact that each algorithm makes at most q queries. Now, we define:

$$S_B := \{(x, h) | h \in S_{B\text{-label}},\ \mathcal{G}^{\mathsf{label}}_{N,m_2}(x) = h\}.$$

Note that, for any iteration, if the adversary successfully guesses S_B in $\hat{S}_{\mathcal{A}}$, then the shared key computed in this iteration would be valid. Specifically, in such a context, there exists an instance of the GGM that is consistent with the query views of both the adversary and the user B, and the validity of the shared key follows by the perfect correctness of $\Pi^{\mathcal{G}_{N,m_2}}$. However, without the knowledge of sk_B, $\mathcal{A}$ might not guess S_B correctly with a good probability. In fact, there are three events:

- Event 1: There exist $(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}$ and $(\mathsf{que}_B, \mathsf{res}_B) \in S_B$ such that $\mathsf{que}_A = \mathsf{que}_B$ but $\mathsf{res}_A \neq \mathsf{res}_B$.
- Event 2: There exist $(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}$ and $(\mathsf{que}_B, \mathsf{res}_B) \in S_B$ such that $\mathsf{que}_A \neq \mathsf{que}_B$ but $\mathsf{res}_A = \mathsf{res}_B$.
- Event 3: For any $(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$, we have that if $\mathsf{que}_A = \mathsf{que}_B$ then $\mathsf{res}_A = \mathsf{res}_B$, and vice versa.

We immediately observe that event 1 occurs at most $6q$ times, because the updating phase would eliminate at least one pair in S_B. Therefore, it suffices to prove that event 2 never occurs except for negligible probability and event 3 would deduce the valid shared key with high probability. According to the description of the adversary Fig. 3, we have that in event 3, the set $\hat{S}_{\mathcal{A}} \cup S_{\textsf{que-res}}$ responds to the labeling queries perfectly and converts the addition queries into labeling queries properly. Concretely, let $\textsf{que} := (h_1, h_2)$ be an addition query, there are two cases: (1) $\hat{S}_{\mathcal{A}} \cup S_{\textsf{que-res}}$ covers $(x_1, h_1), (x_2, h_2)$, and $(x_1 + x_2, h_3)$; (2) either h_1 or h_2 is not stored in $\hat{S}_{\mathcal{A}} \cup S_{\textsf{que-res}}$. For the former case, the response is valid; for latter one, the response is invalid if and only if both h_1 and h_2 are valid group encodings. Therefore, the only bad case that prevents event 3 from deducing the valid shared key is that the adversary outputs a valid group encoding h without knowing the discrete logarithm.

Moreover, in the simulation phase, $\hat{S}_{\mathcal{A}}$ must be consistent with $S_{\textsf{que-res}}$, which indicates that when event 2 occurs, the adversary successfully outputs a valid group encoding h without making labeling query. To bound the probability, we define that, for any $\textsf{sk} \in \mathbb{Z}_N$:

$$Q_{\textsf{sk}} := \left\{(\textsf{que}_1, \textsf{res}_1), \ldots, (\textsf{que}_q, \textsf{res}_q)\right\} \xleftarrow{\textsf{query}} \textsf{KGen}^{\mathcal{G}_{N,m_2}}(\textsf{sk}).$$

Note that the adversary only takes $\textsf{pk}_A$ and $\textsf{pk}_B$ as inputs, where $\textsf{pk}_A = \textsf{KGen}^{\mathcal{G}_{N,m_2}}(\textsf{sk}_A)$ and $\textsf{pk}_B = \textsf{KGen}^{\mathcal{G}_{N,m_2}}(\textsf{sk}_B)$. It is apparent that the group encoding $h \notin S_{\textsf{que-res}}$, and we next establish our analysis by considering the following four cases:

- Case 1: (**Independent group encoding**) $h \notin Q_{\textsf{sk}_A} \cup Q_{\textsf{sk}_B}$
- Case 2: (**Frequent group encoding**) $h \in Q_{\textsf{sk}_A} \cap Q_{\textsf{sk}_B}$
- Case 3: (**Dependent but hard to extract**) $h \in Q_{\textsf{sk}_A} \setminus Q_{\textsf{sk}_B}$.
- Case 4: (**Dependent but hard to extract**) $h \in Q_{\textsf{sk}_B} \setminus Q_{\textsf{sk}_A}$.

It is apparent that, for any query-efficient adversary (might be computationally inefficient), if the probability that it outputs such an h (for all cases) is bounded, then we are done.

Case 1. We note that, h is independent of $\textsf{pk}_A$ and $\textsf{pk}_B$, indicating that the probability that any adversary outputs such a h is bounded by $\frac{O(q)\cdot N}{2^{m_2}} \leq \textsf{negl}(\lambda)$.

Case 2. We first define the frequent group encodings. Specifically, let $t := 26q^2$, we say a group encoding $\textsf{res}$ is frequent if

$$\Pr[(\textsf{que}, \textsf{res}) \in Q_z : z \xleftarrow{\$} \mathbb{Z}_N] \geq \frac{1}{t}.$$

In such a case, we also call $(\textsf{que}, \textsf{res})$ as a frequent query. Note that $\textsf{sk}_A$ and $\textsf{sk}_B$ are uniformly sampled, therefore, for any $(\textsf{que}, \textsf{res}) \in Q_{\textsf{sk}_A}$, if it is not a frequent query, then $\Pr[(\textsf{que}, \textsf{res}) \in Q_{\textsf{sk}_B}] \leq \frac{1}{t}$, indicating that

$$\Pr[Q_{\textsf{sk}_A} \cap Q_{\textsf{sk}_B} \text{ are all frequent queries}] \geq 1 - \frac{q}{t} = 1 - \frac{1}{26q}.$$

Next, we bound the probability that $h \notin S_{\textsf{que-res}}$ conditioned on that $Q_{\textsf{sk}_A} \cap Q_{\textsf{sk}_B}$ are all frequent queries. Let $n := t \cdot \lambda$, we then prove that, with a high probability, $Q_{r_1} \cup \cdots \cup Q_{r_n}$ contains all frequent queries. Essentially, there are at most $q_f := q \cdot t$ frequent queries, denoted as $\{(\textsf{que}'_i, \textsf{res}'_i)\}_{i \in [q_f]}$. For each $(\textsf{que}'_i, \textsf{res}'_i)$, we have that

$$\Pr[(\textsf{que}'_i, \textsf{res}'_i) \notin Q_{r_1} \cup \cdots \cup Q_{r_n}] \leq \left(1 - \frac{1}{t}\right)^n \leq e^{-\lambda},$$

which means

$$\Pr[(\textsf{que}'_i, \textsf{res}'_i) \in Q_{r_1} \cup \cdots \cup Q_{r_n} : \forall i \in [q_f]] \geq 1 - (q \cdot t)e^{-\lambda}.$$

Therefore,

$$\Pr[\text{Case 2}] = \Pr[h \in Q_{\textsf{sk}_A} \cap Q_{\textsf{sk}_B} \wedge h \notin S_{\textsf{que-res}}] \leq \frac{1}{26q} + (q \cdot t)e^{-\lambda} \leq \frac{1}{26q} + \textsf{negl}(\lambda).$$

Case 3. We immediately observe that pk_B is independent of h, which means that only pk_A carries the information of h. Note that the length of pk_A is m_1; in contrast, the length of h is m_2; this intuitively indicates that, over the probability of sampling the GGM instance, it is impossible to extract a valid group encoding in $Q_{\mathsf{sk}_A} \setminus (Q_{\mathsf{sk}_B} \cup S_{\mathsf{que\text{-}res}})$ except for negligible probability.

To establish the formal analysis, we strengthen the adversary $\mathcal{A}$ by providing $\mathcal{A}$ the unbounded computational power, and the following information: the tuple $(\mathsf{sk}_A, \mathsf{sk}_B, \mathsf{pk}_A, Q_{\mathsf{sk}_B}, S_{\mathsf{que\text{-}res}})$. It is easy to see that $\mathcal{A}$ itself can compute $\mathsf{pk}_A, \mathsf{pk}_B$ and $S_{\mathsf{que\text{-}res}}$, therefore it suffices to prove that

$$\Pr[\mathcal{A} \text{ outputs } h \in Q_{\mathsf{sk}_A} \setminus (Q_{\mathsf{sk}_B} \cup S_{\mathsf{que\text{-}res}})] \leq \mathsf{negl}(\lambda)$$

where the probability is over the sampling of $\mathsf{sk}_A, \mathsf{sk}_B$ and the GGM instance[18]. Observe that, pk_B is independent of h, which indicates that knowing sk_B would not increase $\mathcal{A}$'s winning probability. To further simplify the analysis, we prove a more general statement: for any secret key sk and any S (set of query-response tuples, poly-size),

$$\Pr[\mathcal{A}(\mathsf{sk}, \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}), S) \to h : h \in Q_{\mathsf{sk}} \setminus S] \leq \mathsf{negl}(\lambda)$$

where the probability is *only* over the sampling of the GGM instance, conditioned on that the GGM instance $\mathcal{G}_{N,m_2}$ is consistent with S.

Note that, for any fixed poly-size S, the total number of the GGM instances (mapping from N to $\{0,1\}^{m_2}$) that are consistent with S is

$$(2^{m_2} - |S|) \cdot (2^{m_2} - (|S| + 1)) \cdots (2^{m_2} - (N - 1)).$$

Next, we introduce some notations. Note that, once the secret key sk and the GGM instance $\mathcal{G}_{N,m_2}$ are fixed, the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ is deterministic (including the queries made to $\mathcal{G}_{N,m_2}$). We here define $Q_{\mathsf{sk}\text{-}\mathcal{G}}$ as the sequence of the query-response tuples, denoted as

$$Q_{\mathsf{sk}\text{-}\mathcal{G}} := \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}.$$

More clearly, when running the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$, the i-th query that the algorithm makes to $\mathcal{G}_{N,m_2}$ is que_i and the corresponding response is res_i. Besides, for each $(\mathsf{sk}, \mathcal{G}_{N,m_2})$, the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ outputs a public key. Next, we categorize the public keys into two types, namely the "good public keys" and the "bad public keys", with respect to the fixed secret key sk. We denote

$$T = 2^{\frac{m_2 - m_1}{2}} \cdot (2^{m_2} - (|S| + 1)) \cdots (2^{m_2} - (N - 1)),$$

and for any public key pk we denote S_{pk} as the set of the GGM instances such that $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}) = \mathsf{pk}$. Now, we say a public key pk (with respect to sk) is bad if $|S_{\mathsf{pk}}| \leq T$, otherwise we say the public key is good. Note that, given a bad public key pk (e.g., $|S_{\mathsf{pk}}| = 1$), the adversary might output a valid group encoding, thus we need to prove that, over the sampling of the GGM instance,

$$\Pr[\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}) \text{ is bad}] \leq \mathsf{negl}(\lambda).$$

Note that the space of public keys is $\{0,1\}^{m_1}$, which means that there are at most 2^{m_1} public keys. Therefore, the counting of the GGM instances that induce to a bad public key is bounded by $2^{m_1} \times T$, referring to

$$\Pr[\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}) \text{ is bad}] \leq \frac{2^{m_1} \cdot 2^{\frac{m_2 - m_1}{2}}}{(2^{m_2} - |S|)} \leq \frac{1}{2^{\frac{m_2 - m_1}{2}} - |S|} \leq \mathsf{negl}(\lambda).$$

Hence, it suffices to prove that, given any good public key, any adversary $\mathcal{A}$ cannot extract a valid group encoding $h \in Q_{\mathsf{sk}} \setminus S$ except for negligible probability.

[18]The instance of GGM must be consistent with $Q_{\mathsf{sk}_B} \cup S_{\mathsf{que\text{-}res}}$.

For readability, we first elaborate the analysis in the case that $S = \emptyset$, where the adversary only has knowledge of $(\mathsf{sk}, \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}))$. Let str be any string in $\{0,1\}^{m_2}$, we denote S_{str} as the set of GGM instances such that $\mathsf{str} \in Q_{\mathsf{sk}\text{-}\mathcal{G}}$. Therefore it is sufficient to prove that, for any $\mathsf{str} \in \{0,1\}^{m_2}$, the size of S_{str} is much smaller than T (in this special case, $|S| = 0$). Specifically, by having that

$$T > 2^{\frac{m_2 - m_1}{2}} \cdot (2^{m_2} - 1) \cdots (2^{m_2} - (N-1))$$

we prove that

$$|S_{\mathsf{str}}| \leq q \cdot (2^{m_2} - 1) \cdots (2^{m_2} - (N-1))$$

Note that, once the secret key sk and the GGM instance $\mathcal{G}_{N,m_2}$ are fixed, the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ is deterministic. We next illustrate an observation about $Q_{\mathsf{sk}\text{-}\mathcal{G}}$. Let $\mathcal{G}_{N,m_2}$ and $\mathcal{G}'_{N,m_2}$ be two different instances of GGM, and we denote

$$Q_{\mathsf{sk}\text{-}\mathcal{G}} := \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}$$
$$Q_{\mathsf{sk}\text{-}\mathcal{G}'} := \{(\mathsf{que}'_1, \mathsf{res}'_1), \ldots, (\mathsf{que}'_q, \mathsf{res}'_q)\}$$

We claim that either $Q_{\mathsf{sk}\text{-}\mathcal{G}} = Q_{\mathsf{sk}\text{-}\mathcal{G}'}$ or $\exists i \in [q]$ such that $\mathsf{res}_i \neq \mathsf{res}'_i$. In other words, it is impossible that $Q_{\mathsf{sk}\text{-}\mathcal{G}} \neq Q_{\mathsf{sk}\text{-}\mathcal{G}'}$ but $(\mathsf{res}_1, \ldots, \mathsf{res}_q) = (\mathsf{res}'_1, \ldots, \mathsf{res}'_q)$. In fact, if such an event occurs, then there exists an index $j \in [q]$ such that (1) $\forall i < j$, $(\mathsf{que}_i, \mathsf{res}_i) = (\mathsf{que}'_i, \mathsf{res}'_i)$; (2) $\mathsf{que}_j \neq \mathsf{que}'_j$, which contradicts to that $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ is deterministic.

This observation illustrates that $Q_{\mathsf{sk}\text{-}\mathcal{G}}$ can be represented only by $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$; that is, once the sequence of the responses is fixed, then the corresponding sequence of the queries is also settled down. We denote

$$V = ((2^{m_2} - q) \cdots (2^{m_2} - (N-1)))$$

and note that for each response sequence $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$, there are exactly V numbers of GGM instances that would induce it.

Next, we compute the upper bound of $|S_{\mathsf{str}}|$. If str appears in the sequence $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$, then there exists an index i such that $\mathsf{res}_i = \mathsf{str}$. For the rest, we maximize the possibility and have that the number of all possible sequences that contain str is bounded by

$$q \cdot ((2^{m_2} - 1) \cdots (2^{m_2} - (q-1))).$$

Combining the above together, we have that

$$|S_{\mathsf{str}}| \leq q \cdot (2^{m_2} - 1) \cdots (2^{m_2} - (N-1)).$$

In the following, we extend our analysis into the general case, where S is poly-size and

$$T = 2^{\frac{m_2 - m_1}{2}} \cdot (2^{m_2} - (|S| + 1)) \cdots (2^{m_2} - (N-1))$$

We immediately observe that, the upper bound above does not serve our purpose any more. The reason is that the upper bound above is calculated over all possible GGM instances, while what we need to count are the ones over the GGM instances that are consistent with S.

It is apparent that $Q_{\mathsf{sk}\text{-}\mathcal{G}}$ can be still represented by the sequence of responses when $S \neq \emptyset$. To complete the analysis, we then illustrate an additional observation about $Q_{\mathsf{sk}\text{-}\mathcal{G}}$. Let $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$ and $(\mathsf{res}'_1, \ldots, \mathsf{res}'_q)$ be two different sequences. We claim it is impossible that there exists an index $j \in [q]$ such that(1) $\forall i < j$, $\mathsf{res}_i = \mathsf{res}'_i$; (2)$\mathsf{res}_j \in S$ but $\mathsf{res}'_j \notin S$[19]. More specifically, given the statement that $\forall i < j$, $\mathsf{res}_i = \mathsf{res}'_i$, it is apparent that $\mathsf{que}_j = \mathsf{que}'_j$. Moreover, by having $(\mathsf{que}_j, \mathsf{res}_j) \in S$, we claim that the response of que'_j must be res_j, because the GGM instances must be consistent with S. Based on this new observation, we next prove the upper bound by induction.

[19]We here abuse the notation $\mathsf{res}_j \in S$ by meaning that there exists a query/response tuple in S with the response res_j

Let str be a string such that $\mathsf{str} \notin S$ (note that the adversary's goal is to output a valid group encoding without knowing the discrete logarithm), we denote $S_{\mathsf{str}\text{-}k}$ as the set of the GGM instances such that: (1) the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\cdot)$ makes k queries; (2) $\mathsf{str} \in Q_{\mathsf{sk}\text{-}\mathcal{G}} \setminus S$. We then prove that for any k,

$$|S_{\mathsf{str}\text{-}k}| \le k \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)).$$

We first compute $|S_{\mathsf{str}\text{-}1}|$. Note that que_1 is always fixed, and if $\mathsf{que}_1 \in S$[20], then $|S_{\mathsf{str}\text{-}1}| = 0$ because str would never appear. On the other hand, if $\mathsf{que}_1 \notin S$, then the response must be str because str appears. Thus, the counting of the GGM instances that are consistent with $S \cup \{(\mathsf{que}_1, \mathsf{str})\}$ is

$$1 \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))$$

Note that, the response of que_1 is str if and only if those GGM instances are sampled. Moreover, based on our second observation, we have that, either $\mathsf{res}_1 \in S$ or $\mathsf{res}_1 \notin S$. Hence,

$$|S_{\mathsf{str}\text{-}1}| \le \max\{0, 1 \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))\}.$$

Next, given the assumption that

$$|S_{\mathsf{str}\text{-}i}| \le i \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)),$$

we prove

$$|S_{\mathsf{str}\text{-}(i+1)}| \le (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)),$$

Again, que_1 is always fixed, and if $\mathsf{que}_1 \in S$, then $|S_{\mathsf{str}\text{-}(i+1)}|$ is bounded by $|S_{\mathsf{str}\text{-}i}|$, because the response of que_1 is always fixed by S, and str must appear in the last i queries. Thus, it suffices to prove that $|S_{\mathsf{str}\text{-}(i+1)}|$ is properly bounded when $\mathsf{que}_1 \notin S$. Next we consider two scenarios:

- Scenario 1: $\mathsf{res}_1 = \mathsf{str}$;
- Scenario 2: $\mathsf{res}_1 \neq \mathsf{str}$.

Observe that scenario 1 occurs if and only if the GGM instances that are consistent with $S \cup \{(\mathsf{que}_1, \mathsf{str})\}$ are selected. Therefore, the counting of those GGM instances is:

$$1 \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)).$$

When scenario 2 occurs, there are at most $2^{m_2} - (|S|+1)$ options for res_1. Once the response of que_1 is fixed, say $(\mathsf{que}_1, \mathsf{str}')$, we apply the induction. Specifically, we denote $S' = S \cup \{(\mathsf{que}_1, \mathsf{str}')\}$ $(|S'| = |S|+1)$. Note that scenario 2 occurs means that str appears in the last i queries conditioned on that all the GGM instances are consistent with S'. Applying the assumption, we have that the counting of the GGM instances is bounded by

$$\begin{aligned} &(2^{m_2} - (|S|+1)) \cdot i \cdot (2^{m_2} - (|S'|+1)) \cdots (2^{m_2} - (N-1)) \\ =&i \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)). \end{aligned}$$

Now, we see that, if $\mathsf{que}_1 \notin S$ (combining both scenario 1 and scenario 2), then

$$|S_{\mathsf{str}\text{-}(i+1)}| \le (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)).$$

Again, res_1 is either in S or not in S. We have that

$$\begin{aligned} |S_{\mathsf{str}\text{-}(i+1)}| &\le \max\{|S_{\mathsf{str}\text{-}i}|, (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))\} \\ &= (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)) \end{aligned}$$

[20]We here abuse the notation $\mathsf{que}_1 \in S$ by meaning that there exists a query/response pair in S with the query is que_1

By setting $\mathsf{sk} := \mathsf{sk}_A$ and $S := S_{\text{que-res}}$, we have that the probability that the adversary outputs $h \in Q_{\mathsf{sk}_A} \setminus S_{\text{que-res}}$ is bounded by $\frac{O(q^2)}{2^{\frac{m_2-m_1}{2}}} \leq \mathsf{negl}(\lambda)$.

Case 4. It is trivial that

$$\Pr[\text{Case 4}] = \Pr[\text{Case 3}].$$

Combining together, we have that

$$\begin{aligned}\Pr[\mathcal{A}^{\mathcal{G}_{N,m_2}} \text{ outputs the valid shared key}] &\geq 1 - (6q+1)(\frac{1}{26q} + \mathsf{negl}(\lambda)) \\ &\geq \frac{2}{3} - \mathsf{negl}(\lambda).\end{aligned}$$

□

4 The Hierarchy of GGMs

In this section, we establish a hierarchy among GGMs, varying in distinct lengths of group encodings and prove that the shorter GGM is strictly stronger than the longer GGM. Specifically, we show that one can construct an indifferentiable longer generic group from a shorter one plus an addtional independent random oracle, but the shorter generic group model is computationally indifferentiably separated from the longer generic group (when the gap between the lengths is sufficiently large).

4.1 $\mathcal{G}_{N,m_1}$ statistically implies $\mathcal{G}_{N,m_2}$

In this section, we show how to build an longer indifferentiable generic group model from a shorter one plus an additional independent ROM. Here are the building blocks:

- $\mathcal{G}_{N,m_1} := (\mathcal{G}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1})$ is a generic group model that maps $\mathbb{Z}_N$ to $\{0,1\}^{m_1}$;
- $\mathcal{E} : \{0,1\}^{m_2} \to \{0,1\}^{m_2}$ is a random permutation oracle with its inverse $\mathcal{E}^{-1}$.

For simplicity, we denote $\mathcal{O}$ as the tuple $(\mathcal{G}_{N,m_1}, (\mathcal{E}, \mathcal{E}^{-1}))$. The following is the construction $\Pi^{\mathcal{O}}_{\text{L-GGM}} := (L^{\mathcal{O}}_{\text{L-GGM}}, A^{\mathcal{O}}_{\text{L-GGM}})$, depicted in Fig. 4. Correctness easily follows, and it rests to prove the indifferentiability. Formally,

Construction $\Pi^{\mathcal{O}}_{\text{L-GGM}}$

```
$L^{\mathcal{O}}_{\text{L-GGM}}(x)$:
    $\tilde{h} \leftarrow \mathcal{G}^{\mathsf{label}}_{N,m_1}(x)$;
    $h \leftarrow \mathcal{E}(\tilde{h}||0^{(m_2-m_1)})$;
    return $h$.

$A^{\mathcal{O}}_{\text{L-GGM}}(h_1, h_2)$:
    if $\mathcal{E}^{-1}(h_1) \neq \tilde{h}_1||0^{(m_2-m_1)}$: return $\perp$;
    if $\mathcal{E}^{-1}(h_2) \neq \tilde{h}_2||0^{(m_2-m_1)}$: return $\perp$;
    if $\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2) = \perp$: return $\perp$;
    return $\mathcal{E}(\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2)||0^{(m_2-m_1)})$.
```

Figure 4: The construction $\Pi^{\mathcal{O}}_{\text{L-GGM}}$ in the $\mathcal{G}_{N,m_1}$ and RPM.

Theorem 5. *Let m_1, m_2 be two integers that $m_2 \geq m_1$. The scheme $\Pi^{\mathcal{O}}_{\text{L-GGM}}$ in Fig. 4, with access to a generic group $\mathcal{G}_{N,m_1}$, a random permutation $\mathcal{E}$ and its inverse $\mathcal{E}^{-1}$, is indifferentiable from a generic group $\mathcal{G}_{N,m_2}$. More precisely, there exists a simulator $\mathcal{S}$ such that for all $(q_{\mathcal{G}^{\mathsf{label}}_{N,m_1}}, q_{\mathcal{G}^{\mathsf{add}}_{N,m_1}}, q_{\mathcal{E}}, q_{\mathcal{E}^{-1}})$-query distinguisher $\mathcal{D}$ with $q_{\mathcal{G}^{\mathsf{label}}_{N,m_1}} + q_{\mathcal{G}^{\mathsf{add}}_{N,m_1}} + q_{\mathcal{E}} + q_{\mathcal{E}^{-1}} \leq q$, we have*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi^{\mathcal{O}}_{\text{L-GGM}}, \mathcal{G}_{N,m_2}, \mathcal{S}, \mathcal{D}} \leq \frac{6q^2}{N} + \frac{10q^2+4q}{2^{m_1}} + \frac{3q}{2^{\lambda}} + \frac{2q}{2^{m_1}-2q}.$$

The simulator makes at most $3q$ queries to $\mathcal{G}_{N,m_2}$.

Proof Sketch. Correctness of our scheme easily follows. For the indifferentiability, we build a simulator in Fig. 5 to simulate the adversary interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_1}$, $\mathcal{G}^{\mathsf{add}}_{N,m_1}$, $\mathcal{E}$ and $\mathcal{E}^{-1}$ properly. We immediately observe that, our simulator makes at most λq queries to $(\mathcal{G}^{\mathsf{label}}_{N,m_2}, \mathcal{G}^{\mathsf{add}}_{N,m_2})$, and it maintains four tables and the size of each table is at most $2q$, meaning that $\mathcal{S}$ is efficient. In the following, we present the intuitive idea of why $\mathcal{S}$ works. Note that, $\mathcal{E}$ is an ideal random permutation and $\mathcal{G}_{N,m_1}$ is a generic group, hence the responses of a proper simulator should follow these rules:

- Rule 1: The responses of $\mathcal{G}^{\mathsf{label}}_{N,m_1}$ are statistically uniform in $\{0,1\}^{m_1}$;
- Rule 2: There do not exist $x_1 \neq x_2 \in \mathbb{Z}_N$ such that $\mathcal{G}^{\mathsf{label}}_{N,m_1}(x_1) = \mathcal{G}^{\mathsf{label}}_{N,m_1}(x_2)$;
- Rule 3: $\mathcal{G}^{\mathsf{label}}_{N,m_1}(x_1 + x_2) = \mathcal{G}^{\mathsf{add}}_{N,m_1}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x_1), \mathcal{G}^{\mathsf{label}}_{N,m_1}(x_2))$;
- Rule 4: if $\tilde{h} \notin \{\mathcal{G}^{\mathsf{label}}_{N,m_1}(x)\}_{x\in\mathbb{Z}_N}$, then $\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}, \cdot) = \perp$;
- Rule 5: The responses of $\mathcal{E}, \mathcal{E}^{-1}$ are statistically close to a random permutation;
- Rule 6: There is no $r_1 \neq r_2 \in \{0,1\}^{m_2}$ such that $\mathcal{E}(r_1) = \mathcal{E}(r_2)$;
- Rule 7: $\mathcal{E}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x)||0^{(m_2-m_1)}) = \mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$;
- Rule 8: if $\mathcal{G}^{\mathsf{label}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) \neq \perp$, then $\mathcal{E}^{-1}(h) = \tilde{h}||0^{(m_2-m_1)}$.

Next, we illustrate why and how $\mathcal{S}$ achieves those eight rules. Observe that Rule 1 and Rule 7 trivially hold.

Rule 2. The only way to break this rule is if a collision occurs. As $\tilde{h}$ is uniformly sampled, this bad event is trivially bounded by $\frac{2q^2}{2^{m_1}}$.

Rule 3. There are three cases as follows:

- Case 1: The addition query of $\tilde{h}_1, \tilde{h}_2$ has already been put into T_{add};
- Case 2: $\tilde{h}_1, \tilde{h}_2$ have already been put into T_{label};
- Case 3: At least one of $\tilde{h}_1, \tilde{h}_2$ is the unknown encoding.

For Case 1, we note that this equation holds for free.

For Case 2, $\tilde{h}_1, \tilde{h}_2$ are valid encoding and T_{label} holds the corresponding group elements h_1, h_2 in $\mathcal{G}_{N,m_2}$. The simulator $\mathcal{S}$ makes a query $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h_1, h_2)$ with response h, and then looks up h in T_{label}. If h has been in T_{label}, this equation holds for free; otherwise, $\mathcal{S}$ responds to the query with a random $\tilde{h} \in \{0,1\}^{m_1}$ and records $\tilde{h}$. In Case 2, the only way to break this rule is if a collision of $\tilde{h}$ occurs. As $\tilde{h}$ is uniformly sampled, this bad event is trivially bounded by $\frac{2q^2}{2^{m_1}}$.

For Case 3, the simulator $\mathcal{S}$ samples k_1, k_2 from the Bernoulli distribution with parameter $\frac{N}{2^{m_1}}$, and then considers the unknown encoding $\tilde{h}_1$ and/or $\tilde{h}_2$ to be valid if $k_1 = 1$ and/or $k_2 = 1$, respectively. Note that, this probability $\frac{N}{2^{m_1}}$ is close to the probability that a unknown encoding is valid in GGM. For each considered valid encoding, $\mathcal{S}$ randomly samples its pre-image in $\mathbb{Z}_N$, and then makes a query $\mathcal{G}^{\mathsf{label}}_{N,m_2}$ to obtain its corresponding group elements in $\mathcal{G}_{N,m_2}$. After that, the simulator $\mathcal{S}$ responds to the query using table T_{label} or uniformly random sampling $\tilde{h} \in \{0,1\}^{m_1}$, similar to Case 2. In Case 3, there are two bad events that break the rule: 1) for the considered valid encoding, a collision of the random pre-image occurs, which is bounded by $\frac{2q^2}{N}$; 2) for the random response, a collision of the encoding occurs, which is bounded by $\frac{2q^2}{2^{m_1}}$.

Simulator $\mathcal{S}$

```
$\mathcal{S}.\mathcal{G}^{\mathsf{label}}_{N,m_1}(x)$:
 $h \leftarrow \mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$;
 if $\exists(x, \tilde{h}, h) \in T_{\mathsf{label}}$: return $\tilde{h}$;
 if $\exists(\diamond, \tilde{h}, h) \in T_{\mathsf{label}}$: replace $\diamond$ by $x$, return $\tilde{h}$; // $\diamond$ is a symbol.
 $\tilde{h} \xleftarrow{\$} \{0,1\}^{m_1}$, $T_{\mathsf{label}} \leftarrow T_{\mathsf{label}} \cup \{(x, \tilde{h}, h)\}$, $T_{\mathcal{E}} \leftarrow T_{\mathcal{E}} \cup \{(\tilde{h}||0^{(m_2-m_1)}, h)\}$;
 return $\tilde{h}$.

$\mathcal{S}.\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2)$:
 if $\exists(\tilde{h}_1, \tilde{h}_2, \tilde{h}) \in T_{\mathsf{add}} \vee \exists(\tilde{h}_2, \tilde{h}_1, \tilde{h}) \in T_{\mathsf{add}}$: return $\tilde{h}$;
 if $\tilde{h}_1 \in T_\perp \vee \tilde{h}_2 \in T_\perp$: return $\perp$;
 sample $k_1, k_2$ from the Bernoulli distribution with probability $\frac{N}{2^{m_1}}$ for 1;
 for $b \in \{1,2\}$, if $\nexists(*, \tilde{h}_b, y_b) \in T_{\mathsf{label}}$: // $*$ refers to $x \in \mathbb{Z}_N$ or the symbol $\diamond$.
   if $k_b = 0$:
     $h \xleftarrow{\$} \{0,1\}^{m_2}$, if $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) \neq \perp$: resample $h$ (up to $\lambda$ times);
     $T_\perp \leftarrow T_\perp \cup \{\tilde{h}_b\}$, $T_{\mathcal{E}} \leftarrow T_{\mathcal{E}} \cup \{(\tilde{h}||0^{(m_2-m_1)}, h)\}$;
   else:
     $x_b \xleftarrow{\$} \mathbb{Z}_N$, $T_{\mathsf{label}} \leftarrow T_{\mathsf{label}} \cup \{(x_b, \tilde{h}_b, \mathcal{G}^{\mathsf{label}}_{N,m_2}(x_b))\}$, $T_{\mathcal{E}} \leftarrow T_{\mathcal{E}} \cup \{(\tilde{h}||0^{(m_2-m_1)}, h)\}$;
 if $\exists(*, \tilde{h}_1, h_1), (*, \tilde{h}_2, h_2) \in T_{\mathsf{label}}$:
   $h \leftarrow \mathcal{G}^{\mathsf{add}}_{N,m_2}(h_1, h_1)$;
   if $\exists(*, \tilde{h}, h) \in T_{\mathsf{label}}$: $T_{\mathsf{add}} \leftarrow T_{\mathsf{add}} \cup \{(\tilde{h}_1, \tilde{h}_2, \tilde{h})\}$, return $\tilde{h}$;
   $\tilde{h} \xleftarrow{\$} \{0,1\}^{m_1}$, $T_{\mathsf{label}} \leftarrow T_{\mathsf{label}} \cup \{(\diamond, \tilde{h}, h)\}$, $T_{\mathcal{E}} \leftarrow T_{\mathcal{E}} \cup \{(\tilde{h}||0^{(m_2-m_1)}, h)\}$;
   $T_{\mathsf{add}} \leftarrow T_{\mathsf{add}} \cup \{(\tilde{h}_1, \tilde{h}_2, \tilde{h})\}$, return $\tilde{h}$;
 return $\perp$.

$\mathcal{S}.\mathcal{E}(r)$:
 if $\exists(r, h) \in T_{\mathcal{E}}$: return $h$;
 sample $k$ from the Bernoulli distribution with the probability $\frac{N}{2^{m_1}}$ for 1;
 if $r = \tilde{h}||0^{(m_2-m_1)} \wedge k = 1$:
   $x \xleftarrow{\$} \mathbb{Z}_N$, $h \leftarrow \mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$, $T_{\mathsf{label}} \leftarrow T_{\mathsf{label}} \cup \{(x, \tilde{h}, h)\}$;
   $T_{\mathcal{E}} \leftarrow T_{\mathcal{E}} \cup \{(r, h)\}$, return $h$;
 if $x = \tilde{h}||0^{(m_2-m_1)} \wedge k = 0$: $T_\perp \leftarrow T_\perp \cup \{\tilde{h}\}$
 $h \xleftarrow{\$} \{0,1\}^{m_2}$, if $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) \neq \perp$: resample $h$ (up to $\lambda$ times);
 $T_{\mathcal{E}} \leftarrow T_{\mathcal{E}} \cup \{(r, h)\}$, return $h$;

$\mathcal{S}.\mathcal{E}^{-1}(h)$:
 if $\exists(r, h) \in T_{\mathcal{E}}$: return $r$;
 $\tilde{h} \xleftarrow{\$} \{0,1\}^{m_1}$;
 if $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) \neq \perp$: $T_{\mathsf{label}} \leftarrow T_{\mathsf{label}} \cup \{(\diamond, \tilde{h}, h)\}$, $r \leftarrow \tilde{h}||0^{(m_2-m_1)}$;
 sample $k$ from the Bernoulli distribution with the probability $\frac{2^{m_1}-N}{2^{m_2}-N}$ for 1;
 if $k = 1$: $T_\perp \leftarrow T_\perp \cup \{\tilde{h}\}$, $r \leftarrow \tilde{h}||0^{(m_2-m_1)}$;
 else: $t \xleftarrow{\$} \{0,1\}^{(m_2-m_1)} \setminus \{0^{(m_2-m_1)}\}, r \leftarrow x||t$;
 $T_{\mathcal{E}} \leftarrow T_{\mathcal{E}} \cup \{(r, h)\}$, return $r$
```

Figure 5: The simulator for construction $\Pi^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}$.

Rule 4. Note that, for a unknown encoding, the simulator considers it to be group element in $\mathcal{G}_{N,m_1}$ according to the Bernoulli distribution with parameter $\frac{N}{2^{m_1}}$. Hence the probability of the adversary sampling a valid encoding under simulator is close to the one in the generic group $\mathcal{G}_{N,m_1}$.

Rule 5. For $\mathcal{E}$ query, say $\mathcal{E}(r)$, the simulator responds in three cases:

- if there exists a tuple $(r, h) \in T_{\mathcal{E}}$, output h;
- if $r = \tilde{h}||0^{(m_2-m_1)}$, uniformly random sample $x \in \mathbb{Z}_N$ and output $h \leftarrow \mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$ with the probability $\frac{N}{2^{m_1}}$;
- otherwise, uniformly random sample $h \in \{0,1\}^{m_2}$ such that h is not a group element in $\mathcal{G}_{N,m_2}$ via rejection sampling (up to λ times) and output h.

For Case 1, we not that each h stored in table $T_{\mathcal{E}}$ is from generic group $\mathcal{G}_{N,m_2}$ so that it is uniformly random, except for some items generated in the other two cases of $\mathcal{E}$-query. For Case 2, by our construction, $\mathcal{E}$ maps $x = \tilde{h}||0^{(m_2-m_1)}$ to a long group element if $\tilde{h}$ is a valid labeling in $\mathcal{G}_{N,m_1}$. As the probability of adversary sampling a valid encoding in $\mathcal{G}_{N,m_1}$ is close to $\frac{N}{2^{m_1}}$ and $\mathcal{G}_{N,m_2}$ is generic group, $h = \mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$ for random x is uniform and the proper answer. The only way to break this rule is a collision of x occurs, which is trivially bounded by $\frac{2q^2}{N} \leq \mathsf{negl}(\lambda)$. For Case 3, the only way to break this rule in this case is if 1) the response h is a valid group element or 2) a collision of h occurs. As $(m_2 - m_1) \geq 1$ and $2^{m_1} \geq N$, the probability of sampling a valid labeling in $\mathcal{G}_{N,m_2}$ is $\frac{N}{2^{m_2}} \leq \frac{2^{m_1}}{2^{m_2}} \leq \frac{1}{2}$. Due to the rejection sampling up to λ times, the first bad event is bounded by $\dfrac{1}{2^\lambda} \leq \mathsf{negl}(\lambda)$. Easy to note that, the bad event of h collision is trivially bounded by $\frac{2q^2}{2^{m_2}} \leq \mathsf{negl}(\lambda)$. Therefore, in all three cases above, the responses of $\mathcal{E}$ are well-distributed.

For $\mathcal{E}^{-1}$ query, say $\mathcal{E}^{-1}(h)$, the simulator also responds in three case:

- if there exists a tuple $(r, h) \in T_{\mathcal{E}}$, output r;
- if h is a group element in $\mathcal{G}_{N,m_2}$, uniformly random sample $\tilde{h} \in \{0,1\}^{m_1}$ and output $\tilde{h}||0^{(m_2-m_1)}$;
- otherwise, uniformly random sample $\tilde{h} \in \{0,1\}^{m_1}, t \in \{0,1\}^{(m_2-m_1)}$. Then output $\tilde{h}||0^{(m_2-m_1)}$ with the probability $\frac{2^{m_1}-N}{2^{m_2}-N}$; while output $\tilde{h}||t$ with probability $1 - \frac{2^{m_1}-N}{2^{m_2}-N}$.

For Case 1, except for some items from the other two cases of $\mathcal{E}^{-1}$-query, each r stored in table $T_{\mathcal{E}}$ is generated in $\mathcal{G}_{N,m_1}$ queries and corresponds to a group element in $\mathcal{G}_{N,m_1}$. By our construction, such r has the form of a random string $\tilde{h} \in \{0,1\}^{m_1}$ appending $0^{(m_2-m_1)}$ and the bad event is a collision of $\tilde{h}$ occurs, which is trivially bounded by $\frac{2q^2}{2^{m_1}} \leq \mathsf{negl}(\lambda)$. For Case 2, with the same reason of Case 1, $\tilde{h}||0^{(m_2-m_1)}$ is a proper answer. For Case 3, we note that there are 2^{m_1} strings in $\{0,1\}^{m_2}$ ending in $0^{(m_2-m_1)}$, where N strings correspond to valid group elements. Hence given a non-group element h, the probability of its pre-image in $\mathcal{E}$ ending in $0^{(m_2-m_1)}$ is close to $\frac{2^{m_1}-N}{2^{m_2}-N}$. The only way to break this rule in this case is if a collision of r occurs, which is trivially bounded by $\frac{2q^2}{2^{m_1}} \leq \mathsf{negl}(\lambda)$. Therefore, in all three cases above, the responses of $\mathcal{E}^{-1}$ are well-distributed.

Rule 5 and Rule 6. Easy to note that, the only chance that $\mathcal{A}$ violates these rules is bad events in Rule 4 occurs, referring to Rule 5 and Rule 6 holds as long as Rule 4 holds.

Rule 8. Note that if the adversary makes a query $\mathcal{E}^{-1}$ with a group element h, the response's form is well by the Case 3 for $\mathcal{E}^{-1}$ query in Rule 5. Therefore, Rule 8 holds as long as the Case 3 for $\mathcal{E}^{-1}$ query in Rule 5 holds.

Below, we give a full proof of Theorem 5.

Proof. According to the definition of indifferentiability, the adversary has two honest interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_2}$ and $\mathcal{G}^{\mathsf{add}}_{N,m_2}$ and four adversarial interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_1}$, $\mathcal{G}^{\mathsf{add}}_{N,m_1}$, $\mathcal{E}$ and $\mathcal{E}^{-1}$. Therefore, we need to build an efficient simulator $\mathcal{S}$ in the ideal world that can simulate four adversarial interfaces properly, which means, for any PPT differentiator $\mathcal{D}$, the view of $\mathcal{D}$ in the real game is computationally close to the view in the ideal game. We will go through a sequence of hybrid games where in each game, there exists a system that responds to all of the queries (both honest and adversarial) in a slightly different way and then we build our simulator $\mathcal{S}$ as the system in the last game. Before the description of the games, we first specify some parameters:

- There are six types of queries: the labeling query for $\mathcal{G}_{N,m_1}$ $(x, \mathsf{label}; \mathcal{G}_{N,m_1})$, the addition query for $\mathcal{G}_{N,m_1}$ $(\tilde{h}_1, \tilde{h}_2, \mathsf{add}; \mathcal{G}_{N,m_1})$, the labeling query for $\mathcal{G}_{N,m_2}$ $(x, \mathsf{label}; \mathcal{G}_{N,m_2})$, the addition query for $\mathcal{G}_{N,m_2}$ $(h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, the forward $\mathcal{E}$-query $(r, \mathsf{fwd}; \mathcal{E})$ and the inverse $\mathcal{E}^{-1}$-query $(h, \mathsf{inv}; \mathcal{E})$, where $x \in Z_N$, $\tilde{h}_1, \tilde{h}_2 \in \{0,1\}^{m_1}$ and $r, h, h_1, h_2 \in \{0,1\}^{m_2}$.
- The adversary only makes q queries to the system, where $q = \mathsf{poly}(\lambda)$.
- The oracles used in the real world are a long generic group construction $\tilde{\mathcal{G}}_{N,m_2} := (\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}, \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2})$, a short generic group $\tilde{\mathcal{G}}_{N,m_1} := (\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}, \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1})$ and an ideal random permutation $(\tilde{\mathcal{E}}, \tilde{\mathcal{E}}^{-1})$.
- In each game, the system's responses are denoted as $\mathcal{G}^{\mathsf{labelr}}_{N,m_2}$, $\mathcal{G}^{\mathsf{addr}}_{N,m_2}$, $\mathcal{G}^{\mathsf{labelr}}_{N,m_1}$, $\mathcal{G}^{\mathsf{addr}}_{N,m_1}$, $\mathcal{E}^{\mathsf{r}}$ and $\mathcal{E}^{-1\mathsf{r}}$. For instance, $\mathcal{E}^{-1\mathsf{r}}(h)$ denotes the system's response when adversary makes a query $\mathsf{que} := (h, \mathsf{inv}; \mathcal{E})$.

The hybrid games are as follows.

Game 0. This game is identical to the real game except that the system maintains four tables, referring to T_{label}, T_{add}, $T_{\mathcal{E}}$ and $T_{\perp}$. Specifically, the system responds every queries by real oracles. For the tables, the system maintains them as follows:

- T_{label}: It is initiated empty and consists of tuples with form of $(x, \tilde{h}, h)$ or $(\diamond, \tilde{h}, h)$, where $\diamond$ is a symbol for the unknown discrete logarithm. Once the adversary makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_1})$, which does not exist in T_{label} yet, the system inserts $(x, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x), \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x))$ into T_{label}.
- T_{add}: It is initiated empty and consists of tuples with form of $(\tilde{h}_1, \tilde{h}_2, \tilde{h})$. Once the adversary makes a query $(\tilde{h}_1, \tilde{h}_2, \mathsf{add}; \mathcal{G}_{N,m_1})$, which does not exist in T_{label} yet, the system inserts $(\tilde{h}_1, \tilde{h}_2, \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2))$ into T_{add}.
- $T_{\mathcal{E}}$: It is initiated empty and consists of tuples with form of (r, h). Once the adversary makes a query $(r, \mathsf{fwd}; \mathcal{E})$ or $(h, \mathsf{inv}; \mathcal{E})$, which does not exist in $T_{\mathcal{E}}$ yet, the system inserts $(r, \tilde{\mathcal{E}}(r))$ or $(\tilde{\mathcal{E}}^{-1}(h), h)$ into $T_{\mathcal{E}}$.
- $T_{\perp}$: It is initiated empty and consists of element with form of $\tilde{h}$.

Note that all tables are completely hidden to the adversary, and hence the view in real game is identical to the one in Game 0, which refers to

$$\Pr[\text{Game Real}] = \Pr[\text{Game 0}]$$

Then, we illustrate a game that the system responds to part of the queries, by only using the tables and the real oracles. For ease of exposition, we have define a relation between the query que and the tables. Specifically, if que is a labeling query, say $\mathsf{que} := (x, \mathsf{label}; \mathcal{G}_{N,m_1})$, we say $\mathsf{que} \in T_{\mathsf{label}}$ if there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_1 = x$. Analogously, for the addition query $\mathsf{que} := (\tilde{h}_1, \tilde{h}_2, \mathsf{add}; \mathcal{G}_{N,m_1})$, we say $\mathsf{que} \in T_{\mathsf{add}}$ if there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{add}}$ such that $\{t_1, t_2\} = \{\tilde{h}_1, \tilde{h}_2\}$, and we say $\mathsf{que} \in T_{\perp}$ if there exist an element $t \in T_{\perp}$ such that $t = \tilde{h}_1$ or $t = \tilde{h}_2$; for the $\mathcal{E}$-query $\mathsf{que} = (r, \mathsf{fwd}; \mathcal{E})$, we say $\mathsf{que} \in T_{\mathcal{E}}$ if there exists a tuple $(t_1, t_2) \in T_{\mathcal{E}}$ such that $t_1 = r$; for the $\mathcal{E}^{-1}$-query $\mathsf{que} = (h, \mathsf{inv}; \mathcal{E})$, we say $\mathsf{que} \in T_{\mathcal{E}}$ if there exists a tuple $(t_1, t_2) \in T_{\mathcal{E}}$ such that $t_2 = h$.

Game 1. This game is identical to Game 0 except the way of maintaining the tables and responding to the queries. Specifically,

Labeling query. Suppose $\mathsf{que}_k = (x, \mathsf{label}; \mathcal{G}_{N,m_1})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{label}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_1 = x$, then the system responds to the query with t_2.
- Case 2: If $\mathsf{que}_k \notin T_{\mathsf{label}}$, the system first makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_2})$ with the response h, then
  - Case 2.1: If there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_3 = h$, then the system responds to the query with t_2 and replaces t_1 with x.
  - Case 2.2: Otherwise, the system responds with $\tilde{h} \leftarrow \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)$, inserts $(\tilde{h}||0^{(m_2-m_1)}, h)$ to $T_{\mathcal{E}}$, and inserts $(x, \tilde{h}, h)$ into T_{label}.

Addition query. Suppose $\mathsf{que}_k = (\tilde{h}_1, \tilde{h}_2, \mathsf{add}; \mathcal{G}_{N,m_1})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{add}}$ such that $\{t_1, t_2\} = \{\tilde{h}_1, \tilde{h}_2\}$, then the system responds to the query with t_3.
- Case 2: If $\mathsf{que}_k \in T_{\perp}$, then the system responds to the query with $\perp$.
- Case 3: If there exist two tuples $(t_1, t_2, t_3), (t_1', t_2', t_3') \in T_{\mathsf{label}}$ such that $t_2 = \tilde{h}_1, t_2' = \tilde{h}_2$ and $t_1 + t_1' \in T_{\mathsf{label}}$, then the system responds with the corresponding record.
- Case 4: If there exist two tuples $(t_1, t_2, t_3), (t_1', t_2', t_3') \in T_{\mathsf{label}}$ such that $t_2 = \tilde{h}_1, t_2' = \tilde{h}_2$ but $t_1 + t_1' \notin T_{\mathsf{label}}$, then the system makes a query $(t_3, t_3', \mathsf{add}; \mathcal{G}_{N,m_2})$ to obtain h. Then:
  - Case 4.1: If $h \in T_{\mathsf{label}}$, the system responds with the corresponding record.
  - Case 4.2: Otherwise, the system responds with $\tilde{h} \leftarrow \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(t_1 + t_1')$, and inserts $(\diamond, \tilde{h}, h)$ into T_{label} and $(\tilde{h}||0^{(m_2-m_1)}, h)$ into $T_{\mathcal{E}}$.
- Case 5: Otherwise, the system responds to the query with $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2)$, inserts $(\tilde{h}_1, \tilde{h}_2, \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2))$ into T_{add}, and inserts $(\tilde{h}_1||0^{(m_2-m_1)}, \tilde{\mathcal{E}}(\tilde{h}_1))$ and $(\tilde{h}_2||0^{(m_2-m_1)}, \tilde{\mathcal{E}}(\tilde{h}_2))$ into $T_{\mathcal{E}}$. Furthermore, if it holds $\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \mathcal{G}^{\mathsf{label}}_{N,m_1}(1)) = \perp$, the system inserts $\tilde{h}_1$ into $T_{\perp}$; else if $\tilde{h}_1$ does not exist in T_{label} yet, the system inserts $(\diamond, \tilde{h}_1, \tilde{\mathcal{E}}(\tilde{h}_1))$ into T_{label}. Analogously, if $\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_2, \mathcal{G}^{\mathsf{label}}_{N,m_1}(1)) = \perp$, the system inserts $\tilde{h}_2$ into $T_{\perp}$; else if $\tilde{h}_2$ does not exist in T_{label}, the system inserts $(\diamond, \tilde{h}_2, \tilde{\mathcal{E}}(\tilde{h}_2))$ into T_{label}.

$\mathcal{E}$-query. Suppose $\mathsf{que}_k = (r, \mathsf{fwd}; \mathcal{E})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathcal{E}}$, which means there exists a tuple $(t_1, t_2) \in T_{\mathcal{E}}$ such that $t_1 = r$, then the system responds to the query with t_2.
- Case 2: If $\mathsf{que}_k \notin T_{\mathcal{E}}$ and $r = \tilde{h}||0^{(m_2-m_1)}$, the system responds with $h \leftarrow \tilde{\mathcal{E}}(r)$ and inserts (r, h) into $T_{\mathcal{E}}$. Furthermore, if $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(1)) = \perp$, the system inserts $\tilde{h}$ into $T_{\perp}$; else the system inserts $(\diamond, \tilde{h}, h)$ into T_{label}.
- Case 3: Otherwise, the system responds with $h \leftarrow \tilde{\mathcal{E}}(r)$ and inserts (r, h) into $T_{\mathcal{E}}$.

$\mathcal{E}^{-1}$-query. Suppose $\mathsf{que}_k = (h, \mathsf{inv}; \mathcal{E})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathcal{E}}$, which means there exists a tuple $(t_1, t_2) \in T_{\mathcal{E}}$ such that $t_2 = h$, then the system responds to the query with t_1.
- Case 2: If $\mathsf{que}_k \notin T_{\mathcal{E}}$ and $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(1)) \neq \perp$, the system responds with $r \leftarrow \tilde{\mathcal{E}}^{-1}(h)$, parses $r = \tilde{h}||0^{(m_2-m_1)}$, and inserts (r, h) into $T_{\mathcal{E}}$ and $(\diamond, \tilde{h}, h)$ into T_{label}.
- Case 3: Otherwise, the system responds with $r \leftarrow \tilde{\mathcal{E}}^{-1}(h)$ and insert (r, h) into $T_{\mathcal{E}}$. Furthermore, when $r = \tilde{h}||0^{(m_2-m_1)}$, the system inserts $\tilde{h}$ into $T_{\perp}$.

In Game 1, the system keeps longer tables, and for part of the adversarial queries, the system responds to them only using the tables and accessing to the honest interfaces. Note that, each item stored in tables is consistent with the real oracles, these responses are identical to those by calling real oracles. Moreover, in either games, the honest interfaces always correspond to the real oracles. Hence, in either game, the response of any query is identical, which refers to

$$\Pr[\text{Game 0}] = \Pr[\text{Game 1}]$$

Now, the system needs to answer the rest adversary queries by real oracles. In the following hybrid games, we replace the responses that answered by real oracles with tables and honest interfaces without changing the view significantly.

Game 2. This game is identical to Game 1 except for responding to the $\mathcal{E}^{-1}$-queries. Suppose $\mathsf{que}_k = (r, \mathsf{fwd}; \mathcal{E})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathcal{E}}$, then same as in Game 1.
- Case 2: If $\mathsf{que}_k \notin T_{\mathcal{E}}$ and $r = \tilde{h}||0$, if $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(\tilde{h}, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(1)) = \perp$, the system responds with uniformly sampled $h \in \{0,1\}^{m_2}$ such that $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(1)) = \perp$ via reject sampling (up to λ times), and inserts (r, h) into $T_{\mathcal{E}}$ and $\tilde{h}$ into $T_{\perp}$; else the system randomly samples $x \in \mathbb{Z}_N$, responds with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x)$, and inserts (r, h) into $T_{\mathcal{E}}$ and $(\diamond, \tilde{h}, h)$ into T_{label}.
- Case 3: Otherwise, the system responds with uniformly sampled $h \in \{0,1\}^{m_2}$ such that $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(1)) = \perp$ via reject sampling (up to λ times), and inserts (r, h) into $T_{\mathcal{E}}$.

Note that, the only difference between Game 1 and Game 2 occurs in Case 2 and Case 3, where m is previously unknown. In Case 2, which means $r = \tilde{h}||0$, we note that if $\tilde{h}$ is a valid encoding in $\tilde{\mathcal{G}}_{N,m_1}$, $\mathcal{E}^{\mathsf{r}}(r)$ should be a group element in $\tilde{\mathcal{G}}_{N,m_2}$. In fact, the discrete log of each group element in $\tilde{\mathcal{G}}_{N,m_2}$ is independent of the adversary's view. Thus, it is okay to replace $\tilde{\mathcal{E}}(r)$ with $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x)$ of random $x \in \mathbb{Z}_N$, as long as x never appears before que_k. The probability of x-collision is bounded as

$$\Pr[\mathsf{Bad}_1] \leq \frac{2q}{N}.$$

For the situation of invalid $\tilde{h}$ in Case 2 and for Case 3, by the definition, $\mathcal{E}^{\mathsf{r}}(r)$ should be an invalid encoding in $\tilde{\mathcal{G}}_{N,m_1}$. Since all bits of $\tilde{\mathcal{E}}(r)$ are independent of the adversary's view, under the condition that the bad event never happens, the adversary's view on que_k are distributed identically in both games, i.e., $\tilde{\mathcal{E}}(r)$ and random string h. Hence it suffice to prove that except with a negligible probability, the condition is hold. There are two bad events: 1) h appears before que_k; 2) h is a valid group element in $\tilde{\mathcal{G}}_{N,m_2}$. For h-collision event, since h is uniformly sampled from $\{0,1\}^{m_2}$ and h is not a valid group element, we have

$$\Pr[\mathsf{Bad}_2] \leq \frac{2q}{2^{m_2} - N} \leq \frac{2q}{2^{m_2} - 2^{m_1}} \leq \frac{2q}{2^{m_1}}.$$

For the other bad event, it is trivially bounded by

$$\Pr[\mathsf{Bad}_3] \leq (\frac{N}{2^{m_2}})^{\lambda} \leq (\frac{2^{m_1}}{2^{m_2}})^{\lambda} \leq \frac{1}{2^{\lambda}}.$$

Therefore, we have

$$|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \leq q \cdot \sum_{3}^{i=1} \Pr[\mathsf{Bad}_i] \leq \frac{2q^2}{N} + \frac{2q^2}{2^{m_1}} + \frac{q}{2^{\lambda}}.$$

Game 3. This game is identical to Game 2 except for modifying the responses of $\mathcal{E}$ queries once more. Suppose $\mathsf{que}_k = (r, \mathsf{fwd}; \mathcal{E})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathcal{E}}$, then same as in Game 2.
- Case 2: If $\mathsf{que}_k \notin T_{\mathcal{E}}$ and $r = \tilde{h}||0$, the system samples k from the Bernoulli distribution with the probability $\frac{N}{2^{m_1}}$, if $k = 0$, the system responds with uniformly sampled $h \in \{0,1\}^{m_2}$ such that $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(1)) = \perp$ via reject sampling (up to λ times), and inserts (r, h) into $T_{\mathcal{E}}$ and $\tilde{h}$ into $T_\perp$; else the system randomly samples $x \in \mathbb{Z}_N$, responds with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x)$, and inserts (r, h) into $T_{\mathcal{E}}$ and $(\diamond, \tilde{h}, h)$ into T_{label}.
- Case 3: Otherwise, same as in Game 2.

The only difference between Game 2 and Game 3 occurs in Case 2. In Game 2, the system learns whether $\tilde{h}$ is valid by calling $\tilde{\mathcal{G}}_{N,m_1}$; while in Game 3, the system determines it via the Bernoulli distribution with the probability $\frac{N}{2^{m_1}}$. We observe that, for a previously unknown encoding $\tilde{h}$, it is valid in $\tilde{\mathcal{G}}_{N,m_1}$ with the probability $\frac{N-|T_{\mathsf{label}}|}{2^{m_1}-|T_{\mathsf{label}}|-|T_\perp|} \geq \frac{p-2q}{2^{m_1}-2q}$. Therefore, the adversary's view are distributed closely in both games, referring to

$$|\Pr[\text{Game 2}] - \Pr[\text{Game 3}]| \leq \frac{N}{2^{m_1}} - \frac{N-2q}{2^{m_1}-2q} \leq \frac{2q \cdot (2^{m_1} - N)}{2^{m_1} \cdot (2^{m_1} - 2q)} \leq \frac{2q}{2^{m_1}}.$$

Game 4. This game is identical to Game 3 except for responding $\mathcal{E}^{-1}$ queries. Suppose $\mathsf{que}_k = (h, \mathsf{inv}; \mathcal{E})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathcal{E}}$, then same as in Game 3.
- Case 2: If $\mathsf{que}_k \notin T_{\mathcal{E}}$ and $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(1)) \neq \perp$, the system randomly samples $\tilde{h} \in \{0,1\}^{m_1}$, responds with $r \leftarrow \tilde{h}||0^{(m_2-m_1)}$, and inserts (r, h) into $T_{\mathcal{E}}$ and $(\diamond, \tilde{h}, h)$ into T_{label}.
- Case 3: Otherwise, the system first samples $\tilde{h} \xleftarrow{\$} \{0,1\}^{m_1}$ and k from the Bernoulli distribution with the probability $\frac{2^{m_1}-N}{2^{m_2}-N}$, then
  - Case 3.1: If $k = 1$, the system responds with $r \leftarrow \tilde{h}||0^{(m_2-m_1)}$, and inserts (r, h) into $T_{\mathcal{E}}$ and $\tilde{h}$ into $T_\perp$.
  - Case 3.2: Otherwise, the system samples $r \xleftarrow{\$} \{0,1\}^{(m_2-m_1)} \backslash \{0^{(m_2-m_1)}\}$, responds with $r \leftarrow \tilde{h}||t$, and inserts (r, h) into $T_{\mathcal{E}}$.

The differences between Game 2 and Game 3 occurs in Case 2 and Case 3, where h is previously unknown. In Case 2, h is a group element in $\mathcal{G}_{N,m_2}$, hence its pre-image of $\mathcal{E}$ should match the form $\tilde{h}||0^{(m_2-m_1)}$. Note that, it is satisfied in both games. Furthermore, to be consistent, $\tilde{h}$ would be a group element in $\mathcal{G}_{N,m_1}$ and it has the same discrete log as h. All bits each group element in $\tilde{\mathcal{G}}_{N,m_1}(x)$ are independent of the adversary's view. Hence, the adversary's views of $\tilde{\mathcal{G}}_{N,m_1}(x)$ and $\tilde{h}$ are distributed identically, as long as the random string $\tilde{h}$ never appears before. This bad event is bounded as

$$\Pr[\mathsf{Bad}_1] \leq \frac{2q}{2^{m_1}}.$$

For Case 3, which means h is an invalid encoding, the system in Game 2 responses with $\tilde{\mathcal{E}}^{-1}(h)$; while the system in Game 3 responses with random $\tilde{h} \in \{0,1\}^{m_1}$ appended $0^{(m_2-m_1)}$ or random $t \neq 0^{(m_2-m_1)}$, determined by the Bernoulli distribution sample. We observe that, in Game 2, the last $(m_2 - m_1)$ bits of the response are all 0 with the probability $\frac{2^{m_1}-N-|T_\perp|}{2^{m_2}-N-|T_\perp|} \geq \frac{2^{m_1}-N-2q}{2^{m_2}-N-2q}$, which is close to the probability $\frac{2^{m_1}-N}{2^{m_2}-N}$ in

Game 3. Concretely,

$$\begin{aligned}\frac{2^{m_1}-N}{2^{m_2}-N}-\frac{2^{m_1}-N-|T_\perp|}{2^{m_2}-N-|T_\perp|} &\le \frac{2^{m_1}-N}{2^{m_2}-N}-\frac{2^{m_1}-N-2q}{2^{m_2}-N-2q}\\ &= \frac{2q\cdot(2^{m_2}-2^{m_1})}{(2^{m_2}-N-2q)(2^{m_2}-N)}\\ &\le \frac{2q}{2^{m_1}-2q}.\end{aligned}$$

The only bad event in this case is that the response appears before que_k, which is trivially bounded by

$$\Pr[\mathsf{Bad}_2] \le \frac{2q}{2^{m_1}}.$$

Therefore, we have

$$|\Pr[\text{Game 3}]-\Pr[\text{Game 4}]| \le q\cdot\sum_{i=1}^{2}\Pr[\mathsf{Bad}_i]+\frac{2q}{2^{m_1}-2q} \le \frac{4q^2}{2^{m_1}}+\frac{2q}{2^{m_1}-2q}.$$

Game 5. This game is identical to Game 4 except for responding addition queries. Suppose $\mathsf{que}_k = (\tilde{h}_1, \tilde{h}_2, \mathsf{add}; \mathcal{G}_{N,m_1})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, then same as in Game 4.
- Case 2: If $\mathsf{que}_k \in T_\perp$, then same as in Game 4.
- Case 3: If there exist two tuples $(t_1,t_2,t_3),(t_1',t_2',t_3') \in T_{\mathsf{label}}$ such that $t_2=\tilde{h}_1, t_2'=\tilde{h}_2$ and $t_1+t_1' \in T_{\mathsf{label}}$, then same as in Game 4.
- Case 4: If there exist two tuples $(t_1,t_2,t_3),(t_1',t_2',t_3') \in T_{\mathsf{label}}$ such that $t_2=\tilde{h}_1, t_2'=\tilde{h}_2$ but $t_1+t_1' \notin T_{\mathsf{label}}$, then the system makes a query $(t_3,t_3',\mathsf{add};\mathcal{G}_{N,m_2})$ to obtain h. Then:
  - Case 4.1: If $h \in T_{\mathsf{label}}$, then same as in Game 4.
  - Case 4.2: Otherwise, the system responds with uniformly sampled $\tilde{h} \in \{0,1\}^{m_1}$, inserts $(\diamond, \tilde{h}, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(t_3+t_3'))$ into T_{label} and inserts $(\tilde{h}||0^{(m_2-m_1)}, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(t_1+t_1'))$ into $T_{\mathcal{E}}$.
- Case 5: Otherwise, the system samples k_1,k_2 from the Bernoulli distribution with the probability $\frac{N}{2^{m_1}}$ for 1. If $k_1=0$ and $\tilde{h}_1$ does not exist in T_{label}, then the system inserts $\tilde{h}_1$ into $T_\perp$, samples h such that $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(1)) = \perp$ via reject sampling (up to λ times) and inserts $(\tilde{h}_1||0^{(m_2-m_1)}, h)$ into $T_{\mathcal{E}}$; if $k_1=1$ and $\tilde{h}_1$ does not exist in T_{label}, then the system uniformly samples $x\in\mathbb{Z}_N$ as the pre-image of $\tilde{h}_1$, and then inserts $(x, \tilde{h}_1, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x))$ into T_{label} and $(\tilde{h}_1||0^{(m_2-m_1)}, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x))$ into $T_{\mathcal{E}}$. Similarly for k_2 and $\tilde{h}_2$. Finally, the system responds to que_k in the same way as in Case 2, Case 3 or Case 4.

The differences between Game 4 and Game 5 occurs in Case 4.2 and Case 5. Now we analyze the bad event in each case.

In Case 4.2, where $\tilde{h}_1, \tilde{h}_2$ are valid group elements, the system in Game 4 responds with $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(t_2+t_2')$; while the system in Game 5 responds with random string $\tilde{h}\in\{0,1\}^{m_1}$, and records the relationship of $\tilde{h}$ and $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(t_1+t_1')$ in $T_{\mathcal{E}}$. Within the same reason of Game 2 $\approx$ Game 3, the adversary's views of $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(t_2+t_2')$ and $\tilde{h}$ are distributed identically, as long as the random string $\tilde{h}$ never appears before. This bad event is bounded as

$$\Pr[\mathsf{Bad}_1] \le \frac{2q}{2^{m_1}}.$$

In Case 5, where at least one encoding of $\tilde{h}_1, \tilde{h}_2$ never appears before que_k. We observe that, in Game 4, any previously unknown encoding is valid encoding with the probability $\frac{N-|T_{\mathsf{label}}|}{2^{m_1}-|T_{\mathsf{label}}|} \geq \frac{N-2q}{2^{m_1}-2q}$; while in Game 5, the probability is $\frac{N}{2^{m_1}}$ due to the Bernoulli distribution sample. Therefore, under the condition that the bad event never happens, the adversary's view are distributed closely as

$$\frac{N}{2^{m_1}} - \frac{N-|T_{\mathsf{label}}|}{2^{m_1}-|T_{\mathsf{label}}|} \leq \frac{N}{2^{m_1}} - \frac{N-2q}{2^{m_1}-2q} \leq \frac{2q}{2^{m_1}}.$$

Hence it suffice to prove that except with a negligible probability, the condition is hold. Recall that, for any previously unknown $\tilde{h}_b \in \{\tilde{h}_1, \tilde{h}_2\}$ that is (considered) valid, in both games, the system records its corresponding group element in $\tilde{\mathcal{G}}_{N,m_2}$ into T_{label} and $T_{\mathcal{E}}$. In Game 4, the system obtains the group element by calling $\tilde{\mathcal{E}}(\tilde{h}_b||0^{(m_2-m_1)})$; while in Game 5, the system randomly samples $x_b \in \mathbb{Z}_N$ and then calls $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x_b)$. Analogously, the bad event in this situation is that x_b appears before que_k, which is bounded by

$$\Pr[\mathsf{Bad}_2] \leq \frac{2q}{N}.$$

For any previously unknown $\tilde{h}_b \in \{\tilde{h}_1, \tilde{h}_2\}$ that is (considered) invalid, the system inserts it into $T_\perp$ and records its permutation result into $T_{\mathcal{E}}$. Although all bits of each permutation result in $\tilde{\mathcal{E}}$ are independent of the adversary's view, to be consistent, the permutation result of $\tilde{h}_b$ should be an invalid encoding in $\tilde{\mathcal{G}}_{N,m_2}$. Hence, the bad event is that h is a group element in $\tilde{\mathcal{G}}_{N,m_2}$, which is bounded as

$$\Pr[\mathsf{Bad}_3] \leq (\frac{N}{2^{m_2}})^\lambda \leq (\frac{2^{m_1}}{2^{m_2}})^\lambda \leq \frac{1}{2^\lambda}.$$

In addition, if the system finally responds to que_k in the same way as in Case 4.2, it introduces the bad event in Case 4.2 described above.

Thus, we have

$$\begin{aligned}|\Pr[\text{Game 4}] - \Pr[\text{Game 5}]| &\leq q \cdot (\Pr[\mathsf{Bad}_1] + 2\sum_{i=2}^{3} \Pr[\mathsf{Bad}_i]) + \frac{2q}{2^{m_1}} \\ &\leq \frac{2q^2+2q}{2^{m_1}} + \frac{4q^2}{N} + \frac{2q}{2^\lambda}.\end{aligned}$$

Game 6. This game is identical to Game 5 except for responding labeling queries. Suppose $\mathsf{que}_k = (x, \mathsf{label}; \mathcal{G}_{N,m_1})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{label}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T$ such that $t_1 = x$, then same as in Game 5.
- Case 2: If $\mathsf{que}_k \notin T_{\mathsf{label}}$, the system first makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_2})$ with the response h, then
  - Case 2.1: If there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_3 = h$, then same as in Game 5.
  - Case 2.2: Otherwise, the system responds with uniformly sampled $\tilde{h} \in \{0,1\}^{m_1}$, inserts $(\tilde{h}||0^{(m_2-m_1)}, h)$ to $T_{\mathcal{E}}$, and inserts $(x, \tilde{h}, h)$ into T_{label}.

The only difference between Game 5 and Game 6 occurs in Case 2.2, which means $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)$ never appears before que_k. Note that, all bits of $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)$ are distributed uniformly and independent of the adversary's view. Thus, under the condition that the bad event never happens, the adversary's view are distributed identical in both games. The bad event is that $\tilde{h}$ appears before que_k, which is trivially bounded by

$$\Pr[\mathsf{Bad}] \leq \frac{2q}{2^{m_1}}.$$

Therefore, we have

$$|\Pr[\text{Game 5}] - \Pr[\text{Game 6}]| \le q \cdot \Pr[\mathsf{Bad}] \le \frac{2q^2}{2^{m_1}}.$$

Ideal Game. In Game 6, the queries to the adversarial interfaces are answered by the tables which are maintained by the system and by making queries to honest interface. It is straightforward to show that we can replace $\tilde{\mathcal{G}}_{N,m_2}$ with a generic group $\mathcal{G}_{N,m_2}$, resulting in Ideal Game.

The difference between Game 6 and Ideal Game is that: in Game 6, the system responds to all queries by calling $\tilde{\mathcal{G}}_{N,m_2}$; while in Ideal Game, the system makes queries to $\mathcal{G}_{N,m_2}$. In fact, as $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x) = \tilde{\mathcal{E}}(\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)||0^{(m_2-m_1)})$ in which $\tilde{\mathcal{E}}$ is an ideal random permutation and $\tilde{\mathcal{G}}_{N,m_1}$ is a generic group, all bits of $\tilde{\mathcal{G}}_{N,m_2}(x)$ are uniformly random. Analogously, if h_1, h_2 are valid group elements, all bits of $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h_1, h_2)$ are uniformly random. Thus, the distribution of $\tilde{\mathcal{G}}_{N,m_2}$ and $\mathcal{G}_{N,m_2}$ are identical, referring to

$$\Pr[\text{Game 7}] = \Pr[\text{Ideal Game}]$$

In the following, we give the full description of the simulator.

Simulator in the Ideal Game. Let $\mathcal{G}_{N,m_2}$ be a generic group. By definition, the simulator $\mathcal{S}$ has access to the honest interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_2}, \mathcal{G}^{\mathsf{add}}_{N,m_2}$. And for the adversarial queries, $\mathcal{S}$ works as the system in Game 7. Concretely, $\mathcal{S}$ maintains four tables: the labeling table T_{label}, the addition table T_{add}, the permutation table $T_{\mathcal{E}}$ and the invalid table $T_{\perp}$. And $\mathcal{S}$ answers the adversarial queries by the tables and the honest interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_2}, \mathcal{G}^{\mathsf{add}}_{N,m_2}$.
Labeling query. Suppose $\mathsf{que}_k = (x, \mathsf{label}; \mathcal{G}_{N,m_1})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{label}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T$ such that $t_1 = x$, then the simulator responds to the query with t_2.
- Case 2: If $\mathsf{que}_k \notin T_{\mathsf{label}}$, the system first makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_2})$ with the response h, then
  - Case 2.1: If there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_3 = h$, then the simulator responds to the query with t_2 and replaces t_1 with x.
  - Case 2.2: Otherwise, the simulator responds with uniformly sampled $\tilde{h} \in \{0,1\}^{m_1}$, inserts $(\tilde{h}||0^{(m_2-m_1)}, h)$ to $T_{\mathcal{E}}$, and inserts $(x, \tilde{h}, h)$ into T_{label}.

Addition query. Suppose $\mathsf{que}_k = (\tilde{h}_1, \tilde{h}_2, \mathsf{add}; \mathcal{G}_{N,m_1})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{add}}$ such that $\{t_1, t_2\} = \{\tilde{h}_1, \tilde{h}_2\}$, then the simulator responds to the query with t_3.
- Case 2: If $\mathsf{que}_k \in T_{\perp}$, then the simulator responds to the query with $\perp$.
- Case 3: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_2 = \tilde{h}_1, t'_2 = \tilde{h}_2$ and $t_1 + t'_1 \in T_{\mathsf{label}}$, then the simulator responds with the corresponding record.
- Case 4: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_2 = \tilde{h}_1, t'_2 = \tilde{h}_2$ but $t_1 + t'_1 \notin T_{\mathsf{label}}$, then the system makes a query $(t_3, t'_3, \mathsf{add}; \mathcal{G}_{N,m_2})$ to obtain h. Then:
  - Case 4.1: If $h \in T_{\mathsf{label}}$, the simulator responds with the corresponding record.
  - Case 4.2: Otherwise, the simulator responds with uniformly sampled $\tilde{h} \in \{0,1\}^{m_1}$, inserts $(\diamond, \tilde{h}, \mathcal{G}^{\mathsf{label}}_{N,m_2}(t_3 + t'_3))$ into T_{label} and inserts $(\tilde{h}||0^{(m_2-m_1)}, \mathcal{G}^{\mathsf{label}}_{N,m_2}(t_1 + t'_1))$ into $T_{\mathcal{E}}$.
- Case 5: Otherwise, the simulator samples k_1, k_2 from the Bernoulli distribution with the probability $\frac{N}{2^{m_1}}$ for 1. If $k_1 = 0$ and $\tilde{h}_1$ does not exist in T_{label}, then the simulator inserts $\tilde{h}_1$ into $T_{\perp}$, samples h such that $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) = \perp$ via reject sampling (up to λ times) and inserts $(\tilde{h}_1||0^{(m_2-m_1)}, h)$ into $T_{\mathcal{E}}$; if $k_1 = 1$ and $\tilde{h}_1$ does not exist in T_{label}, then the simulator uniformly samples $x \in \mathbb{Z}_N$ as the pre-image

of $\tilde{h}_1$, and then inserts $(x, \tilde{h}_1, \mathcal{G}^{\mathsf{label}}_{N,m_2}(x))$ into T_{label} and $(\tilde{h}_1||0^{(m_2-m_1)}, \mathcal{G}^{\mathsf{label}}_{N,m_2}(x))$ into $T_{\mathcal{E}}$. Similarly for k_2 and $\tilde{h}_2$. Finally, the system responds to que_k in the same way as in Case 2, Case 3 or Case 4.

$\mathcal{E}$-query. Suppose $\mathsf{que}_k = (r, \mathsf{fwd}; \mathcal{E})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathcal{E}}$, which means there exists a tuple $(t_1, t_2) \in T_{\mathcal{E}}$ such that $t_1 = r$, then the simulator responds to the query with t_2.
- Case 2: If $\mathsf{que}_k \notin T_{\mathcal{E}}$ and $m = \tilde{h}||0$, the simulator samples k from the Bernoulli distribution with the probability $\frac{N}{2^{m_1}}$, if $k = 0$, the simulator responds with uniformly sampled $h \in \{0,1\}^{m_2}$ such that $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) = \perp$ via reject sampling (up to λ times), and inserts (r, h) into $T_{\mathcal{E}}$ and $\tilde{h}$ into $T_{\perp}$; else the simulator randomly samples $x \in \mathbb{Z}_N$, responds with $h \leftarrow \mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$, and inserts (r, h) into $T_{\mathcal{E}}$ and $(\diamond, \tilde{h}, h)$ into T_{label}.
- Case 3: Otherwise, the simulator responds with uniformly sampled $h \in \{0,1\}^{m_2}$such that $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) =$ $\perp$ via reject sampling (up to λ times), and inserts (r, h) into $T_{\mathcal{E}}$.

$\mathcal{E}^{-1}$-query. Suppose $\mathsf{que}_k = (h, \mathsf{inv}; \mathcal{E})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathcal{E}}$, which means there exists a tuple $(t_1, t_2) \in T_{\mathcal{E}}$ such that $t_2 = h$, then the simulator responds to the query with t_1.
- Case 2: If $\mathsf{que}_k \notin T_{\mathcal{E}}$ and $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h, \mathcal{G}^{\mathsf{label}}_{N,m_2}(1)) \neq \perp$, the simulator randomly samples $\tilde{h} \in \{0,1\}^{m_1}$, responds with $r \leftarrow \tilde{h}||0^{(m_2-m_1)}$, and inserts (r, h) into $T_{\mathcal{E}}$ and $(\diamond, \tilde{h}, h)$ into T_{label}.
- Case 3: Otherwise, the simulator first samples $\tilde{h} \xleftarrow{\$} \{0,1\}^{m_1}$ and k from the Bernoulli distribution with the probability $\frac{2^{m_1}-N}{2^{m_2}-N}$, then
  - Case 3.1: If $k = 1$, the simulator responds with $r \leftarrow \tilde{h}||0^{(m_2-m_1)}$, and inserts (r, h) into $T_{\mathcal{E}}$ and $\tilde{h}$ into $T_{\perp}$.
  - Case 3.2: Otherwise, the simulator samples $t \xleftarrow{\$} \{0,1\}^{(m_2-m_1)} \setminus \{0^{(m_2-m_1)}\}$, responds with $r \leftarrow \tilde{h}||t$, and inserts (r, h) into $T_{\mathcal{E}}$.

We have shown that every pair of adjacent games are indistinguishable. Combining together, we establish the entire proof, referring to

$$\Big|\Pr[\text{Real Game}] - \Pr[\text{Ideal Game}]\Big| \leq \frac{6q^2}{N} + \frac{10q^2+4q}{2^{m_1}} + \frac{3q}{2^{\lambda}} + \frac{2q}{2^{m_1}-2q} \leq \mathsf{negl}(\lambda).$$

□

4.2 $\mathcal{G}_{N,m_2}$ does not computationally imply $\mathcal{G}_{N,m_1}$

In this section, we show that the shorter GGM is computationally indifferentiably separated from the longer one. Formally,

Theorem 6. *Let λ be the security parameter. Let $\mathcal{G}_{N,m_1}$ and $\mathcal{G}_{N,m_2}$ be two generic group models. If $(m_2 - m_1) \geq \omega(\log \lambda)$, then $\mathcal{G}_{N,m_1}$ is computationally indifferentiably separated from $\mathcal{G}_{N,m_2}$.*

Essentially, our strategy highly relies on the analysis in [ZZ23], and thus we first recap the discrete logarithm identification problem (DLI) with respect to the generic groups.

Definition 11 (DLI w.r.t. generic groups [ZZ23])**.** *Given a generic group $\mathcal{G}_{N,m} := (\mathcal{G}^{\mathsf{label}}_{N,m}, \mathcal{G}^{\mathsf{add}}_{N,m})$, a group element $h \leftarrow \mathcal{G}^{\mathsf{label}}_{N,m}(x)$, where $x \xleftarrow{\$} \mathbb{Z}_N$, an efficient algorithm with access to $\mathcal{G}_{N,m}$ outputs a "query-free" circuit C_{GGM} such that C_{GGM} identifies x with a good probability, specifically,*

- $\Pr[C_{\mathsf{GGM}}(x) = 1] \geq \frac{1}{2}$;
- *for any noticeable function ρ, $\Pr_{x' \neq x}[C_{\mathsf{GGM}}(x') = 1] \leq \rho$,*

where the probability is over the sampling of x' and C_{GGM}'s randomness. We say DLI w.r.t. generic groups is hard if no query-efficient adversary outputs such a circuit.

To absorb Zhandry and Zhang's analysis into our setting, we then propose the DLI problem w.r.t the shorter groups in the longer GGM.

Definition 12 (DLI w.r.t. shorter groups in longer GGM)**.** *Given a shorter group construction $\Pi^{\mathcal{G}_{N,m_2}} := (L^{\mathcal{G}_{N,m_2}}, A^{\mathcal{G}_{N,m_2}})$, whose order is N and the length of group encoding is m_1, in a longer generic group $\mathcal{G}_{N,m_2}$ and a group element $h \leftarrow L^{\mathcal{G}_{N,m_2}}(x)$ where $x \xleftarrow{\$} \mathbb{Z}_N$, an efficient algorithm with access to $\mathcal{G}_{N,m}$ outputs a "query-free" circuit $C_{\mathsf{G\text{-}GGM}}$ such that $C_{\mathsf{G\text{-}GGM}}$ identifies x with a good probability, specifically,*

- $\Pr[C_{\mathsf{G\text{-}GGM}}(x) = 1] \geq \frac{2}{3}$;
- *for any noticeable function ρ, $\Pr_{x' \neq x}[C_{\mathsf{G\text{-}GGM}}(x') = 1] \leq \rho$,*

where the probability is over the sampling of x' and $C_{\mathsf{G\text{-}GGM}}$'s randomness. We say DLI w.r.t. shorter groups in the longer GGM is hard if no query-efficient adversary outputs such a circuit.

Now, we are ready to establish the entire proof of Theorem 6.

Proof. Suppose a construction $\Pi^{\mathcal{G}_{N,m_2}} := (L^{\mathcal{G}_{N,m_2}}, A^{\mathcal{G}_{N,m_2}})$ is indifferentiable from $\mathcal{G}_{N,m_1}$ in the longer GGM $\mathcal{G}_{N,m_2}$. The argument goes in three steps:

1. DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy.
2. If $\Pi^{\mathcal{G}_{N,m_2}}$ is indifferentiable from $\mathcal{G}_{N,m_1}$ and DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy, then DLI w.r.t. $\mathcal{G}_{N,m_1}$ is also easy.
3. Yet, DLI w.r.t. the generic group $\mathcal{G}_{N,m_1}$ is hard.

The above three steps draw a contradiction, so the statement "$\Pi^{\mathcal{G}_{N,m_2}}$ is indifferentiable from $\mathcal{G}_{N,m_1}$" cannot be true, completing our proof. Below, we prove them step by step; for readability, we do them in reverse order.

Lemma 3. *DLI w.r.t. the generic group $\mathcal{G}_{N,m_1}$ is hard.*

The following proof of Lemma 3 is taken verbatim from [ZZ23].

Proof. Assuming there is an efficient adversary $\mathcal{A}$ making at most q queries to $\mathcal{G}_{N,m_1}$ which breaks DLI, we can convert $\mathcal{A}$ into a query-efficient adversary $\mathcal{B}$, as shown in Fig. 6, that breaks Discrete Logarithm in the GGM[21].

According to the description of $\mathcal{B}$, we have that $\mathcal{B}$ makes at most q queries to $\mathcal{G}_{N,m_1}$. We set ρ to be $\frac{1}{q^{12}}$ and mbound to be $1 + \frac{N-1}{q^6}$, respectively; applying Markov's inequality, it is apparent that $\Pr[|S| > \mathsf{mbound}] \leq \frac{1}{q^6}$. Therefore, we have that

$$\begin{aligned}\Pr[\mathcal{B} \text{ outputs } x] &= \sum_{i=1}^{N} \Pr[|S| = i] \cdot (\frac{1}{i}) \cdot \Pr[x \in S \big| |S| = i] \\ &\geq \Pr[|S| \leq \mathsf{mbound}] \cdot (\frac{1}{\mathsf{mbound}}) \cdot \Pr[x \in S \big| |S| \leq \mathsf{mbound}].\end{aligned}$$

[21]Note that $\mathcal{B}$ is time-inefficient if N is super-polynomial, which is not a problem in our setting since the lower bound of discrete log in the GGM [Sho97] only counts the number of oracle queries.

Adversary $\mathcal{B}^{\mathcal{G}_{N,m_1}}$

```
$\mathcal{B}^{\mathcal{G}_{N,m_1}}(\mathcal{G}^{\text{label}}_{N,m_1}(1), \mathcal{G}^{\text{label}}_{N,m_1}(x))$:
$S \leftarrow \emptyset$; $C_{\mathsf{GGM}} \leftarrow \mathcal{A}^{\mathcal{G}_{N,m_1}}(\mathcal{G}^{\text{label}}_{N,m_1}(1), \mathcal{G}^{\text{label}}_{N,m_1}(x))$;
for $i = 0$ to $N-1$,
    if $C_{\mathsf{GGM}}(i) = 1$, $S \leftarrow S \cup \{i\}$;
$y \xleftarrow{\$} S$;
return $y$.
```

Figure 6: A query-efficient adversary breaks Discrete Logarithm in the GGM

Note that $\Pr[x \in S] \geq 1/2$, which means that $\Pr[x \in S \big| |S| \leq \mathsf{mbound}]$ should be close to $1/2$, because $|S| \leq \mathsf{mbound}$ happens with high probability. Formally, we denote A and B to be the events "$x \in S$" and "$|S| \leq \mathsf{mbound}$", respectively, we have that

$$\Pr[\mathsf{A}\big|\mathsf{B}] = \frac{\Pr[\mathsf{A}] - \Pr[\mathsf{A}\big|\neg\mathsf{B}]\Pr[\neg\mathsf{B}]}{\Pr[\mathsf{B}]} \geq \frac{\Pr[\mathsf{A}] - \Pr[\neg\mathsf{B}]}{\Pr[\mathsf{B}]} = 1 - \frac{1-\Pr[\mathsf{A}]}{\Pr[\mathsf{B}]}.$$

Substituting into the inequalities above, we have that,

$$\begin{aligned}\Pr[\mathcal{B}\text{ outputs }x] &\geq (1-\frac{1}{q^6})\cdot(\frac{1}{1+\frac{N-1}{q^6}})\cdot(1-\frac{1}{2}\cdot(1-\frac{1}{q^6}))\\ &\geq \frac{1}{2}\cdot(1-\frac{1}{q^6})\cdot\frac{q^6}{q^6+N-1} = \Theta(\frac{q^6}{N}) > O(\frac{q^2}{N}),\end{aligned}$$

which contradicts the hardness of discrete log in the GGM [Sho97]. □

Lemma 4. *If $\Pi^{\mathcal{G}_{N,m_2}}$ is indifferentiable from $\mathcal{G}_{N,m_1}$ and DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy, then DLI w.r.t. $\mathcal{G}_{N,m_1}$ is also easy.*

Proof. Let $\mathcal{A}$ be an adversary which breaks DLI with respect to $\Pi^{\mathcal{G}_{N,m_2}}$, that is, $\mathcal{A}$ takes $L^{\mathcal{G}_{N,m_2}}(x)$ as input, makes at most q queries to $\mathcal{G}_{N,m_2}$, outputs a query-free circuit $C_{\mathsf{G\text{-}GGM}}$ such that $C_{\mathsf{G\text{-}GGM}}$ identifies x with a good probability. According to the definition of indifferentiability, we know that there exists a simulator $\mathcal{S}$ that makes at most $q^* = \mathsf{poly}(q)$ queries to $\mathcal{G}_{N,m_1}$ and simulates the generic group $\mathcal{G}_{N,m_2}$ properly. That is, in the ideal world, given the adversary $\mathcal{A}$ that takes $\mathcal{G}_{N,m_1}(x)$ as input, $\mathcal{A}^{\mathcal{S}^{\mathcal{G}_{N,m_1}}} = \mathcal{A}^{(\mathcal{S}^{\mathcal{G}_{N,m_1}})}$ outputs a query-free circuit C_{GGM}[22]. By definition, we have that no efficient differentiator $\mathcal{D}$ distinguishes these two circuits, i.e., $C_{\mathsf{G\text{-}GGM}}$ and C_{GGM}. Next, we claim that C_{GGM} also identifies x with a good probability.

C_{GGM} Accepts x with a Good Probability. In this part, we prove that if $\Pr[C_{\mathsf{GGM}}(x) = 1] < \frac{1}{2}$, then there exists an efficient differentiator which breaks indifferentiability.

According to the differentiator in Fig. 7, we have that $\Pr[\mathcal{D}_{\mathsf{real}} = 1] \geq \frac{2}{3}$ but if $\Pr[C_{\mathsf{GGM}}(x) = 1] < \frac{1}{2}$, we have that $\Pr[\mathcal{D}_{\mathsf{ideal}} = 1] < \frac{1}{2}$, which means the advantage of the differentiator in Fig. 7 is at least $\frac{2}{3} - \frac{1}{2} = \frac{1}{6}$.

C_{GGM} Accepts x' ($x' \neq x$) with a Small Probability. In this part, we prove that for any noticeable function ρ, $\Pr_{x' \neq x}[C_{\mathsf{GGM}}(x') = 1] \leq \rho$. By contraposition, we assume that there exists a noticeable function ρ^*, such that $\Pr_{x' \neq x}[C_{\mathsf{GGM}}(x') = 1] > \rho^*$, then we proceed to build an efficient differentiator breaks indifferentiability. Let n be a polynomial such that $n \geq \frac{1}{\rho^*}$, we build our differentiator in Fig. 8.

In the real world, by the statement of Lemma 4, we have that for any noticeable function, say $\frac{0.08}{n}$, $\Pr[C_{\mathsf{GGM}}(z_i) = 1] \leq \frac{0.08}{n}$. According to the differentiator in Fig. 8, immediately observe that $\Pr[\mathcal{D}_{\mathsf{real}} = 1] \leq n \cdot \frac{0.08}{n} = 0.08$, due to union bound.

[22]Note that $\mathcal{A}^{\mathcal{S}}$ is the adversary against DLI with respect to $\mathcal{G}_{N,m_1}$.

```
Differentiator-1

Dreal in real world :
    $x \xleftarrow{\$} \mathbb{Z}_N$;
    $C_{\mathsf{GGM}} \leftarrow \mathcal{A}^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x))$;
    if $C_{\mathsf{GGM}}(x) = 1$, return 1;
    return 0.

Dideal in ideal world :
    $x \xleftarrow{\$} \mathbb{Z}_N$;
    $C_{\mathsf{G\text{-}GGM}} \leftarrow \mathcal{A}^{\mathcal{S}^{\mathcal{G}_{N,m_1}}}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x))$;
    if $C_{\mathsf{G\text{-}GGM}}(x) = 1$, return 1;
    return 0.
```

Figure 7: Differentiator-1 against Indifferentiability.

```
Differentiator-2

Dreal in real world :
    $x \xleftarrow{\$} \mathbb{Z}_N$; $z_1, \ldots, z_n \xleftarrow{\$} \mathbb{Z}_N \setminus \{x\}$;
    $C_{\mathsf{GGM}} \leftarrow \mathcal{A}^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x))$;
    for $i = 1$ to $n$;
        if $C_{\mathsf{GGM}}(z_i) = 1$, then return 0;
    return 1.

Dideal in ideal world :
    $x \xleftarrow{\$} \mathbb{Z}_N$; $z_1, \ldots, z_n \xleftarrow{\$} \mathbb{Z}_N \setminus \{x\}$;
    $C_{\mathsf{G\text{-}GGM}} \leftarrow \mathcal{A}^{\mathcal{S}^{\mathcal{G}_{N,m_1}}}(\mathcal{G}_{N,m_1}(x))$;
    for $i = 1$ to $n$;
        if $C_{\mathsf{G\text{-}GGM}}(z_i) = 1$, then return 0;
    return 1.
```

Figure 8: Differentiator-2 against Indifferentiability.

On the other hand, in the ideal world, if $\Pr_{x' \neq x}[C_{\mathsf{GGM}}(x') = 1] > \rho^*$, then,

$$\Pr[\mathcal{D}_{\mathsf{ideal}} = 1] \geq 1 - (1-\rho^*)^n \geq 1 - (1-\rho^*)^{\frac{1}{\rho^*}}$$
$$\geq 1 - \frac{1}{e} > 0.6 \ .$$

For the first inequality above, $\mathcal{D}_{\mathsf{ideal}} = 1$ means that $\forall i, C_{\mathsf{GGM}}(z_i) = 0$, and due to independence of z_i, it is apparent that

$$\Pr[\mathcal{D}_{\mathsf{ideal}} = 1 | (\forall i, z_i \neq x)] \geq 1 - (1-\rho^*)^n.$$

Therefore, the advantage of the differentiator in Fig. 8 is at least $(0.6 - 0.08) > 0.1$. We conclude that if $\Pi^{\mathcal{G}_{N,m_2}}$ is indifferentiable from $\mathcal{G}_{N,m_1}$ and DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy, then DLI w.r.t. $\mathcal{G}_{N,m_1}$ is easy.

□

Lemma 5. *DLI w.r.t.* $\Pi^{\mathcal{G}_{N,m_2}}$ *is easy.*

Proof. By the definition of indifferentiability, the algorithms $L^{\mathcal{G}_{N,m_2}}$ and $A^{\mathcal{G}_{N,m_2}}$ are deterministic; and they shall support group operations correctly with high probability. More specifically, there exists a negligible function ϵ_1, such that

$$\Pr_{x \neq y}[L^{\mathcal{G}_{N,m_2}}(x) = L^{\mathcal{G}_{N,m_2}}(y)] \leq \epsilon_1, \tag{1}$$
$$\Pr[A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x), L^{\mathcal{G}_{N,m_2}}(y)) = L^{\mathcal{G}_{N,m_2}}(x+y)] \geq 1 - \epsilon_1, \tag{2}$$

where the probability is over the sampling of $x, y \in \mathbb{Z}_N$ and the generic group $\mathcal{G}_{N,m_2}$. As explained above, we stress that $L^{\mathcal{G}_{N,m_2}}$ only makes labeling queries. Let q be an integer in $\mathsf{poly}(\lambda)$. We assume that both $L^{\mathcal{G}_{N,m_2}}$ and $A^{\mathcal{G}_{N,m_2}}$ make at most q queries to $\mathcal{G}_{N,m_2}$. Next, we prove that the DLI problem w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy by constructing an efficient adversary $\mathcal{A}$ and a query-free circuit $C_{\mathsf{G\text{-}GGM}}$ in Fig. 9. (Here, G-GGM denotes the shorter group in the longer GGM.)

We first clarify some undefined notions in Fig. 9. Let n be a sufficiently large integer to be specified below. By $\left\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\right\} \xleftarrow{\mathsf{query}} L^{\mathcal{G}_{N,m_2}}(r_i)$, we denote that on input r_i, the algorithm $L^{\mathcal{G}_{N,m_2}}(r_i)$

```
Adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}$

$\mathcal{A}^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x))$:
$S_{\text{que-res}} \leftarrow \emptyset$; $z, r_1, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_N$;
for $i = 1$ to $n$: //collecting frequent queries
    $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\mathsf{query}} L^{\mathcal{G}_{N,m_2}}(r_i)$;
    $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}$;
compute $L^{\mathcal{G}_{N,m_2}}(z)$, $L^{\mathcal{G}_{N,m_2}}(x-z) \leftarrow A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x), L^{\mathcal{G}_{N,m_2}}(-z))$;
$\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\mathsf{query}} A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z))$;
for $j = 1$ to $q$: //collecting queries in group addition
    if $\mathsf{que}_j$ is an addition query: //converting addition queries into labeling queries
        parse the addition query $\mathsf{que}_j$ to two labels $h_1, h_2$;
        if $\exists (x_1, h_1), (x_2, h_2) \in S_{\text{que-res}}$: $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup \{(x_1 + x_2, \mathsf{res}_j)\}$;
    else: $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup \{(\mathsf{que}_j, \mathsf{res}_j)\}$; //collecting labeling queries
return $C_{\text{G-GGM}}(\,\cdot\,, S_{\text{que-res}}, L^{\mathcal{G}_{N,m_2}}(x))$.

$C_{\text{G-GGM}}(\,\cdot\,, S_{\text{que-res}}, L^{\mathcal{G}_{N,m_2}}(x))$:
take $z \in \mathbb{Z}_N$ as input; run $\mathsf{str} \leftarrow L^{S_{\text{que-res}}}(z)$;
when $L$ makes a labeling query $\mathsf{que} = x$: //responding to labeling queries
    if $\exists (x, h) \in S_{\text{que-res}}$: respond with $h$;
    else: respond with a uniformly sampled $h$; $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup \{(x, h)\}$;
when $L$ makes an addition query $\mathsf{que} = (h_1, h_2)$: //responding to addition queries
    if $\exists (x_1, h_1), (x_2, h_2) \in S_{\text{que-res}}$:
        if $\exists (x_1 + x_2, h) \in S_{\text{que-res}}$: respond with $h$;
        else: respond with a uniformly sampled $h$; $S_{\text{que-res}} \leftarrow S_{\text{que-res}} \cup \{(x_1 + x_2, h)\}$;
    else: respond with $\perp$; //if addition query has a new labeling, then responds with $\perp$
if $\mathsf{str} = L^{\mathcal{G}_{N,m_2}}(x)$: return 1; else: return 0.
```

Figure 9: Efficient Adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}$ and query-free circuit $C_{\text{G-GGM}}$ w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$.

makes queries $(\mathsf{que}_1, \ldots, \mathsf{que}_q)$ to $\mathcal{G}_{N,m_2}$ and gets responses of $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$; and similar for the notation $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\mathsf{query}} A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z))$. [23] Given an input $z \in \mathbb{Z}_N$, the query-free circuit $C_{\text{G-GGM}}$ runs algorithm $L^{\mathcal{G}_{N,m_2}}(z)$ except for replacing the querying oracle by looking up the table $S_{\text{que-res}}$ (and lazy sampling); we denote that as $L^{S_{\text{que-res}}}$.

Next, we give evidence that the query-free circuit $C_{\text{G-GGM}}$ in Fig. 9 identifies x with a good probability, which means DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy. Recall that, we say $C_{\text{G-GGM}}$ identifies x with a good probability if:

- $\Pr[C_{\text{G-GGM}}(x) = 1] \geq \frac{2}{3}$;
- for any noticeable function ρ: $\Pr_{x' \neq x}[C_{\text{G-GGM}}(x') = 1] \leq \rho$.

We prove that $C_{\text{G-GGM}}$ satisfies these two properties one by one.

[23] Here, we abuse the notation $L^{\mathcal{G}_{N,m_2}}(x-z)$ as both the group element and the labeling operation on $x - z$.

$C_{\textsf{G-GGM}}$ **Accepts x with Good Probability.** Here, we prove $\Pr[C_{\textsf{G-GGM}}(x) = 1] \geq \frac{2}{3}$. As shown in Fig. 9, $C_{\textsf{G-GGM}}(x)$ outputs 1 if and only if $L^{S_{\textsf{que-res}}}(x) = L^{\mathcal{G}_{N,m_2}}(x)$. For any $x, y \in \mathbb{Z}_N$, let $S_{x,y}$ be the set of all group elements appeared in responses when invoking $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x), L^{\mathcal{G}_{N,m_2}}(y))$. We denote

$$Q_x := \{(\textsf{que}_1, \textsf{res}_1), \dots, (\textsf{que}_q, \textsf{res}_q)\} \xleftarrow{\textsf{query}} L^{\mathcal{G}_{N,m_2}}(x);$$
$$Q_{x,y} := \{(\textsf{que}, \textsf{res}) | \textsf{res} \in S_{x,y}, \mathcal{G}^{\textsf{label}}_{N,m_2}(\textsf{que}) = \textsf{res}\}.$$

The adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}$ in Fig. 9 collects the queries/responses as

$$S_{\textsf{que-res}} = Q_{r_1} \cup \cdots \cup Q_{r_n} \cup Q_{x-z,z} \text{ except for responses relative to new labels.}$$

Obviously, if $Q_x \subseteq S_{\textsf{que-res}}$, then $L^{S_{\textsf{que-res}}}(x) = L^{\mathcal{G}_{N,m_2}}(x)$. Unfortunately, x is independent of the adversary's view and some queries in Q_x may be ignored when running $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z))$. We cannot guarantee $Q_x \subseteq S_{\textsf{que-res}}$. In fact, for each $(\textsf{que}, \textsf{res}) \in Q_x$, there are four cases as follows:

- Case 1: the representation of $L^{\mathcal{G}_{N,m_2}}(x)$ is independent of $\textsf{res}$;
- Case 2: $\textsf{res}$ does not appear in $Q_{x-z,z}$;
- Case 3: $\textsf{res}$ appears in labeling queries made by $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z))$;
- Case 4: $\textsf{res}$ appears in addition queries made by $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z))$.

For Case 1, we argue the representation of $L^{\mathcal{G}_{N,m_2}}(x)$ is independent of $Q_x \setminus Q_x^*$, where $Q_x^* := Q_x \cap (Q_{x-z} \cup Q_z \cup Q_{x-z,z})$. Concretely, we observe that there are two query-free algorithms, denoted as $\textsf{Alg}_1$ and $\textsf{Alg}_2$, such that $L^{\mathcal{G}_{N,m_2}}(x) = \textsf{Alg}_1(x, Q_x) = \textsf{Alg}_2(x-z, z, Q_{x-z} \cup Q_z \cup Q_{x-z,z})$ with high probability. That is, $\textsf{Alg}_1$ computes the group element through the labeling operation $L^{\mathcal{G}_{N,m_2}}$ and $\textsf{Alg}_2$ computes the same group element through the addition operation $A^{\mathcal{G}_{N,m_2}}$; according to Equation 2, $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z)) = L^{\mathcal{G}_{N,m_2}}(x)$ holds except for a negligible probability ϵ_1. Therefore, the output of $\textsf{Alg}_1(x, Q_x)$ is independent of the group elements in $Q_{x-z} \cup Q_z \cup Q_{x-z,z} \setminus Q_x$; the output of $\textsf{Alg}_2(x-z, z, Q_{x-z} \cup Q_z \cup Q_{x-z,z})$ is independent of the group elements in $Q_x \setminus Q_{x-z} \cup Q_z \cup Q_{x-z,z}$. We conclude that the representation of $L^{\mathcal{G}_{N,m_2}}(x)$ is independent of $Q_x \setminus Q_x^*$. Therefore, the queries we care about are only in Case 2, 3 and 4, and it is sufficient to prove that $Q_x^* \subseteq S_{\textsf{que-res}}$ with a good probability.

Next, we prove it by two steps as follows:

- $Q_x \cap Q_{x-z,z} \subseteq S_{\textsf{que-res}}$ with a good probability;
- $Q_x \cap (Q_z \cup Q_{x-z}) \subseteq S_{\textsf{que-res}}$ with a good probability.

$Q_x \cap Q_{x-z,z} \subseteq S_{\textsf{que-res}}$ *with a good probability.* We immediately observe that, the adversary can easily collect all group elements (with their discrete logarithms) in $Q_{x-z,z}$, except for the group codings that are the responses of addition queries (with new and valid labelings as inputs). The challenge is that there may be valid new labels in addition queries made by $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z))$, denoted as $S_{\textsf{new}}$; but the adversary does not know their discrete logarithms. To address this challenge, we first show that with a noticeable probability all group elements in $S_{\textsf{new}}$ are frequent (defined below), and then collect labeling queries from a sufficiently large number of samples, say $Q_{r_1} \cup \cdots \cup Q_{r_n}$, to capture all the frequent group elements with their discrete logarithms. That is, $S_{\textsf{new}} \subseteq Q_{r_1} \cup \cdots \cup Q_{r_n}$ with a noticeable probability. Here, we define the frequent group elements. Let $t := 200q$, we say a labeling query $(\textsf{que}, \textsf{res})$ and the group element $\textsf{res}$ are frequent, if

$$\Pr[(\textsf{que}, \textsf{res}) \in Q_z : z \xleftarrow{\$} \mathbb{Z}_N] \geq \frac{1}{t}.$$

Applying exactly the same analysis as in Lemma 2, we have that any efficient algorithm $A^{\mathcal{G}_{N,m_2}}$ cannot extract a valid group element of $\mathcal{G}_{N,m_2}$ from single $L^{\mathcal{G}_{N,m_2}}(x-z)$ or $L^{\mathcal{G}_{N,m_2}}(z)$ except for negligible probability. Therefore, we have $S_{\textsf{new}} \subseteq Q_{x-z} \cap Q_z$ with high probability. Because both x and r are uniformly sampled, $x - z$ and r are independent. For any labeling query $(\mathsf{que}, \mathsf{res}) \in Q_{x-z}$, if $(\mathsf{que}, \mathsf{res})$ is not frequent, then

$$\Pr[(\mathsf{que}, \mathsf{res}) \in Q_z] \leq \frac{1}{t}.$$

Thus, we have that

$$\Pr[Q_x \cap Q_{x-z,z} \text{ are all frequent queries}] \geq 1 - \frac{q}{t} = 1 - \frac{1}{200}.$$

So there exists a negligible function ϵ_2 (for $S_{\mathsf{new}} \nsubseteq Q_{x-z} \cap Q_z$), such that

$$\Pr[S_{\mathsf{new}} \text{ are all frequent group elements}] \geq 1 - \frac{1}{200} - \epsilon_2.$$

Let $n := t^2 \cdot q$. Below, we prove that $Q_{r_1} \cup \cdots \cup Q_{r_n}$ contains all frequent queries with a noticeable probability. In fact, there are at most $q_f := q \cdot t$ frequent queries. Denote all frequent queries as $\{(\mathsf{que}'_i, \mathsf{res}'_i)\}_{i \in [q_f]}$. For each $(\mathsf{que}'_i, \mathsf{res}'_i)$, we have

$$\Pr[(\mathsf{que}'_i, \mathsf{res}'_i) \notin Q_{r_1} \cup \cdots \cup Q_{r_n}] \leq \left(1 - \frac{1}{t}\right)^n \leq e^{-q \cdot t},$$

which refers to

$$\Pr[(\mathsf{que}'_i, \mathsf{res}'_i) \in Q_{r_1} \cup \cdots \cup Q_{r_n} : \forall i \in [q_f]] \geq 1 - (q \cdot t)e^{-q \cdot t} \geq 1 - 200e^{-200}.$$

We conclude that, if $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x - z), L^{\mathcal{G}_{N,m_2}}(z)) = L^{\mathcal{G}_{N,m_2}}(x)$, then

$$\Pr[S_{\mathsf{new}} \text{ are all put into } S_{\mathsf{que\text{-}res}}] \geq 1 - 200e^{-200} - \frac{1}{200} - \epsilon_2.$$

$Q_x \cap (Q_z \cup Q_{x-z}) \subseteq S_{\mathsf{que\text{-}res}}$ *with a good probability.* Note that x, z and $x - z$ are pairwise independent. Within the same reason described above,

$$\Pr[Q_x \cap Q_z \text{ are all frequent queries}] \geq 1 - \frac{1}{200};$$
$$\Pr[Q_x \cap Q_{x-z} \text{ are all frequent queries}] \geq 1 - \frac{1}{200}$$

which refers to

$$\Pr[Q_x \cap (Q_z \cup Q_{x-z}) \text{ are all frequent queries}] \geq 1 - \frac{1}{100}.$$

Therefore, we have that

$$\Pr[Q_x \cap (Q_z \cup Q_{x-z}) \text{ are all put into } S_{\mathsf{que\text{-}res}}] \geq 1 - \frac{1}{100} - 200e^{-200}.$$

Combining together, we have that

$$\Pr[Q_x^* \subseteq S_{\mathsf{que\text{-}res}}] \geq 1 - \frac{1}{200} - 400e^{-200} - \frac{1}{100} - \epsilon_2 - \epsilon_1 \geq \frac{2}{3},$$

which refers to $\Pr[C_{\mathsf{G\text{-}GGM}} \text{ accepts } x] \geq \frac{2}{3}$.

$C_{\textsf{G-GGM}}$ **Accepts x' ($x' \neq x$) with Small Probability.** Here, we prove that for any noticeable function ρ, $\Pr_{x' \neq x}[C_{\textsf{G-GGM}}(x') = 1] \leq \rho$. Recall that, $C_{\textsf{G-GGM}}(x')$ outputs 1 if and only if $L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x)$. There are two cases as follows:

- Case 1: $Q_{x'} \subseteq S_{\textsf{que-res}}$ and $L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x)$;
- Case 2: $Q_{x'} \nsubseteq S_{\textsf{que-res}}$ and $L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x)$.

Immediately observe that if $Q_{x'} \subseteq S_{\textsf{que-res}}$, then $L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x')$. According to Equation 1, we have

$$\Pr[\text{Case 1}] \leq \epsilon_1.$$

For Case 2, it is enough to analyze $\Pr[L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x)]$. We first define an event "$S_{\textsf{que-res}}$ is good" if $\Pr_{x' \neq x}[L^{S_{\textsf{que-res}}}(x') = L^{S_{\textsf{que-res}}}(x)] \leq \sqrt{\epsilon_1}$ (the nonce of $S_{\textsf{que-res}}$ is fixed); otherwise we say $S_{\textsf{que-res}}$ is bad. When $S_{\textsf{que-res}}$ is good, for any str $\in \{0,1\}^{m_1}$, $\Pr[L^{S_{\textsf{que-res}}}(x') = \textsf{str}] \leq 2\sqrt[4]{\epsilon_1}$. By Equation 1, we have

$$\begin{aligned}&\Pr_{x' \neq x}[L^{S_{\textsf{que-res}}}(x') = L^{S_{\textsf{que-res}}}(x) | S_{\textsf{que-res}} \text{ is good}] \cdot \Pr[S_{\textsf{que-res}} \text{ is good}] \\ &+ \Pr_{x' \neq x}[L^{S_{\textsf{que-res}}}(x') = L^{S_{\textsf{que-res}}}(x) | S_{\textsf{que-res}} \text{ is bad}] \cdot \Pr[S_{\textsf{que-res}} \text{ is bad}] \leq \epsilon_1\end{aligned}$$

It's apparent that $\Pr[S_{\textsf{que-res}} \text{ is bad}] \leq \sqrt{\epsilon_1}$. Therefore,

$$\begin{aligned}&\Pr[L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x)] \\ =&\Pr[L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x) | S_{\textsf{que-res}} \text{ is good}] \cdot \Pr[S_{\textsf{que-res}} \text{ is good}] \\ &+ \Pr[L^{S_{\textsf{que-res}}}(x') = L^{\mathcal{G}_{N,m_2}}(x) | S_{\textsf{que-res}} \text{ is bad}] \cdot \Pr[S_{\textsf{que-res}} \text{ is bad}] \\ \leq& 2\sqrt[4]{\epsilon_1} + \sqrt{\epsilon_1}.\end{aligned}$$

which refers to $\Pr[\text{Case 2}] \leq 2\sqrt[4]{\epsilon_1} + \sqrt{\epsilon_1}$.

Combining together, we have that

$$\Pr[C_{\textsf{G-GGM}}(x') = 1] \leq \epsilon_1 + 2\sqrt[4]{\epsilon_1} + \sqrt{\epsilon_1}$$

which means that for any noticeable function ρ, $\Pr_{x' \neq x}[C_{\textsf{G-GGM}}(x') = 1] \leq \rho$.

□

Combining together, we establish the entire theorem.

□

4.3 The hierarchy is tight

In this section, we build an indifferentiable $\mathcal{G}_{N,m_1}$ from a longer generic group $\mathcal{G}_{N,m_2}$, for $(m_2 - m_1) = 1$ and $m_1 \geq \log N + \omega(\log \lambda)$. Our construction can be trivially extended to the case that $(m_2 - m_1) \leq \Theta(\log \lambda)$, illustrating that our impossibility result is tight.

In Fig. 10, we present a shorter GGM (S-GGM) construction $\Pi_{\textsf{S-GGM}}^{\mathcal{G}_{N,m_2}} := (L_{\textsf{S-GGM}}^{\mathcal{G}_{N,m_2}}, A_{\textsf{S-GGM}}^{\mathcal{G}_{N,m_2}})$ where $\textsf{Trunc}$ is a function that takes an m_2-bit string as input and outputs its first m_1 bits. For each group operation $L_{\textsf{S-GGM}}^{\mathcal{G}_{N,m_2}}$ or $A_{\textsf{S-GGM}}^{\mathcal{G}_{N,m_2}}$, the corresponding longer label in $\mathcal{G}_{N,m_2}$ is truncated as the result. Specifically, for any input shorter label $\tilde{h}$ in $A_{\textsf{S-GGM}}^{\mathcal{G}_{N,m_2}}$, it is easy to obtain the corresponding longer label h (or determine that $\tilde{h}$ is invalid) through a brute-force approach. That is, call $\mathcal{G}_{N,m_2}$ to identify which of $\tilde{h}||0$ and $\tilde{h}||1$ is a group element. We observe that the bad event is that both $\tilde{h}||0$ and $\tilde{h}||1$ are valid, which is bounded by $\frac{2^{(m_2-m_1)} \cdot N}{2^{m_2}} = \frac{N}{2^{m_1}}$.

Formally, we have the following theorem showing that the $\Pi_{\textsf{S-GGM}}^{\mathcal{G}_{N,m_2}}$ construction is indifferentiable from $\mathcal{G}_{N,m_1}$.

Construction $\Pi_{\text{S-GGM}}^{\mathcal{G}_{N,m_2}}$

```
$L_{\text{S-GGM}}^{\mathcal{G}_{N,m_2}}(x)$:
    return $\mathsf{Trunc}(\mathcal{G}_{N,m_2}(x))$.
```

```
$A_{\text{S-GGM}}^{\mathcal{G}_{N,m_2}}(\tilde{h}_1, \tilde{h}_2)$:
    $h_1 \leftarrow \diamond$; $h_2 \leftarrow \diamond$;
    for $b := 0$ to $1$:
        if $\mathcal{G}^{\text{add}}_{N,m_2}(\tilde{h}_1||b, \mathcal{G}^{\text{label}}_{N,m_2}(1)) \neq \perp$: $h_1 \leftarrow \tilde{h}_1||b$;
        if $\mathcal{G}^{\text{add}}_{N,m_2}(\tilde{h}_2||b, \mathcal{G}^{\text{label}}_{N,m_2}(1)) \neq \perp$: $h_2 \leftarrow \tilde{h}_2||b$;
    if $h_1 = \diamond$ or $h_2 = \diamond$: return $\perp$;
    return $\mathsf{Trunc}(\mathcal{G}^{\text{add}}_{N,m_2}(h_1, h_2))$.
```

Figure 10: Construction $\Pi_{\text{S-GGM}}^{\mathcal{G}_{N,m_2}}$ in $\mathcal{G}_{N,m_2}$, where $(m_2 - m_1) = 1$ and $m_1 \geq \log N + \omega(\log \lambda)$.

Theorem 7. *The construction $\Pi_{\text{S-GGM}}^{\mathcal{G}_{N,m_2}}$ in Fig. 10, with access to a generic group $\mathcal{G}_{N,m_2}$, is indifferentiable from a generic group $\mathcal{G}_{N,m_1}$, where $(m_2 - m_1) = 1$ and $m_1 \geq \log N + \omega(\log \lambda)$. More precisely, there exists a simulator $\mathcal{S}$ such that for all $(q_{\mathcal{G}^{\text{label}}_{N,m_2}}, q_{\mathcal{G}^{\text{add}}_{N,m_2}})$-query distinguisher $\mathcal{D}$ with $q_{\mathcal{G}^{\text{label}}_{N,m_2}} + q_{\mathcal{G}^{\text{add}}_{N,m_2}} \leq q$, we have*

$$\mathrm{Adv}^{\text{indif}}_{\Pi_{\text{S-GGM}}^{\mathcal{G}_{N,m_2}}, \mathcal{G}_{N,m_1}, \mathcal{S}, \mathcal{D}} \leq \frac{(2q+1)N}{2^{m_1}}.$$

The simulator makes at most $3q$ queries to $\mathcal{G}_{N,m_1}$.

Simulator $\mathcal{S}$

```
$\mathcal{S}.\mathcal{G}^{\text{label}}_{N,m_2}(x)$:
    $\tilde{h} \leftarrow \mathcal{G}^{\text{label}}_{N,m_1}(x)$;
    if $\exists (x, \tilde{h}, h) \in T_{\text{label}}$: return $h$;
    if $\exists (\diamond, \tilde{h}, h) \in T_{\text{label}}$: replace $\diamond$ by $x$, return $h$; // $\diamond$ is a symbol.
    $b \xleftarrow{\$} \{0,1\}$, $h \leftarrow \tilde{h}||b$, $T_{\text{label}} \leftarrow T_{\text{label}} \cup (x, \tilde{h}, h)$;
    return $h$.
```

```
$\mathcal{S}.\mathcal{G}^{\text{add}}_{N,m_2}(h_1, h_2)$:
    if $\exists (h_1, h_2, h) \in T_{\text{add}}$ or $\exists (h_2, h_1, h) \in T_{\text{add}}$: return $h$;
    $\tilde{h}_1 \leftarrow \mathsf{Trunc}(h_1)$, $\tilde{h}_2 \leftarrow \mathsf{Trunc}(h_2)$;
    if $(\tilde{h}_1, h_1) \in T_\perp \ \vee \ (\tilde{h}_2, h_2) \in T_\perp$: return $\perp$;
    if $\mathcal{G}^{\text{add}}_{N,m_1}(\tilde{h}_1, \mathcal{G}^{\text{label}}_{N,m_1}(1)) = \perp \ \vee \ \mathcal{G}^{\text{add}}_{N,m_1}(\tilde{h}_2, \mathcal{G}^{\text{label}}_{N,m_1}(1)) = \perp$: return $\perp$;
    for $i \in \{1,2\}$, if $\nexists (*, \tilde{h}_i, h_i) \in T_{\text{label}}$: // $*$ refers to $x \in \mathbb{Z}_N$ or symbol $\diamond$.
        if $\exists (*, \tilde{h}_i, h'_i) \in T_{\text{label}} \ \wedge \ h_i \neq h'_i$: return $\perp$;
        sample $k_i$ from the Bernoulli distribution with the probability $\frac{1}{2-|T_\perp[\tilde{h}_i]|}$ for 1;
        if $k_i = 0$: $T_\perp \leftarrow T_\perp \cup \{(\tilde{h}_i, h_i)\}$, return $\perp$;
        else: $T_{\text{label}} \leftarrow T_{\text{label}} \cup \{(\diamond, \tilde{h}_i, h_i)\}$;
    $\tilde{h} \leftarrow \mathcal{G}^{\text{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2)$;
    if $\exists (*, \tilde{h}, h) \in T_{\text{label}}$: $T_{\text{add}} \leftarrow T_{\text{add}} \cup \{(h_1, h_2, h)\}$, return $h$;
    $b \xleftarrow{\$} \{0,1\}$, $h \leftarrow \tilde{h}||b$, $T_{\text{label}} \leftarrow T_{\text{label}} \cup \{(\diamond, \tilde{h}, h)\}$;
    $T_{\text{add}} \leftarrow T_{\text{add}} \cup \{(h_1, h_2, h)\}$, return $h$.
```

Figure 11: The simulator for construction $\Pi_{\text{S-GGM}}^{\mathcal{G}_{N,m_2}}$.

Proof Sketch. The correctness of our scheme easily follows. For the indifferentiability, the adversary has two honest interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_1}$ and $\mathcal{G}^{\mathsf{add}}_{N,m_1}$ and two adversarial interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_2}$, $\mathcal{G}^{\mathsf{add}}_{N,m_2}$. We build a simulator in Fig. 11 to simulate the adversary interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_2}$ and $\mathcal{G}^{\mathsf{add}}_{N,m_2}$ properly. In the following, we give the high-level intuition of our proof strategy.

Our simulator makes at most $3q$ queries to $(\mathcal{G}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1})$, and it maintains three tables $T_{\mathsf{label}}, T_{\mathsf{add}}, T_{\perp}$ each of size at most $2q$, meaning that $\mathcal{S}$ is efficient. Note that, the notation $T_{\perp}[\tilde{h}_i]$ in Fig. 11 stands for a subset of $T_{\perp}$ whose items are indexed by $\tilde{h}_i$, formally, $T_{\perp}[\tilde{h}_i] := \{(t_1,t_2)|(t_1,t_2) \in T_{\perp} \wedge t_1 = \tilde{h}_i\}$. Next, we present the intuitive idea of why $\mathcal{S}$ works. Note that, $\mathcal{G}_{N,m_2}$ is a generic group, hence the responses of a proper simulator should follow the these rules:

- Rule 1: The responses of $\mathcal{G}^{\mathsf{label}}_{N,m_2}$ are statistically uniform in $\{0,1\}^{m_2}$;
- Rule 2: There is no $x_1 \neq x_2 \in \mathbb{Z}_N$ such that $\mathcal{G}^{\mathsf{label}}_{N,m_2}(x_1) = \mathcal{G}^{\mathsf{label}}_{N,m_2}(x_2)$;
- Rule 3: $\mathcal{G}^{\mathsf{label}}_{N,m_1}(x) = \mathsf{Trunc}(\mathcal{G}^{\mathsf{label}}_{N,m_2}(x))$;
- Rule 4: $\mathcal{G}^{\mathsf{label}}_{N,m_2}(x_1+x_2) = \mathcal{G}^{\mathsf{add}}_{N,m_2}(\mathcal{G}^{\mathsf{label}}_{N,m_2}(x_1), \mathcal{G}^{\mathsf{label}}_{N,m_2}(x_2))$.
- Rule 5: if $h \notin \{\mathcal{G}^{\mathsf{label}}_{N,m_2}(x)\}_{x\in\mathbb{Z}_N}$, then $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h,\cdot) = \perp$.

Next, we illustrate how our simulator $\mathcal{S}$ achieves five rules above. Observe that the rule 1, 2 and 3 trivially hold.

Rule 4. Denote $h_1 := \mathcal{G}^{\mathsf{label}}_{N,m_2}(x_1), h_2 := \mathcal{G}^{\mathsf{label}}_{N,m_2}(x_2)$. There are four cases:

- Case 1: There exists $(h_1,h_2,h) \in T_{\mathsf{add}}$;
- Case 2: $h_1, h_2 \in T_{\mathsf{label}}$, and T_{label} holds the corresponding addition result;
- Case 3: $h_1, h_2 \in T_{\mathsf{label}}$, but T_{label} does not have the addition result;
- Case 4: Otherwise, at least one of h_1, h_2 is previously unknown.

Obviously, for Case 1 and 2, this equation holds for free. For Case 3, from the short generic group $\mathcal{G}_{N,m_1}$, the adversary can only learn the first m_1 bits of the long addition result. Therefore, it is okay to replace the last one bit by a random bit $b \in \{0,1\}$. The only way to break this rule is that a collision of h occurs, which never happens due to the first m_1 bits from $\mathcal{G}_{N,m_1}$.

Now we analyze Case 4. We immediate observe that, if the first m_1 bits of the previously unknown encoding, denoted as $\tilde{h}_i := \mathsf{Trunc}(h_i)$, is not a valid group element in $\mathcal{G}_{N,m_1}$, then h_i is invalid due to our construction and our simulator always responds with $\perp$. Otherwise, there are two subcases as follows: 1) there is a valid encoding $h' \in T_{\mathsf{label}}$ such that $\mathsf{Trunc}(h') = \tilde{h}_i$; 2) T_{label} doesn't have such h' yet. In the first subcase, h_i is always an invalid encoding unless both $\tilde{h}||0$ and $\tilde{h}||1$ are valid group elements in the generic group $\mathcal{G}_{N,m_2}$, which is bounded by a negligible probability $\frac{N}{2^{m_1}} \leq \frac{1}{2^{\omega(\log\lambda)}}$. Therefore, it is okay of our simulator to always responds with $\perp$. In the second subcase, the simulator considers h_i to be a valid group element according to the Bernoulli distribution with the probability $\frac{1}{2-T_{\perp}[\tilde{h}_i]}$. Easy to note that, the probability of the adversary sampling a valid group element under our simulator is close to the one under the generic group $\mathcal{G}_{N,m_2}$. When all previously unknown encodings are considered valid, our simulator responds to the addition query with $\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1,\tilde{h}_2)||b$, where b is uniformly sampled from $\{0,1\}$. Within the same reason of Case 3, Rule 4 holds in Case 4.

Rule 5. There are two cases as follows:

- Case 1: the encoding h is already put into $T_{\perp}$;
- Case 2: the encoding h is previously unknown.

Note that, the simulator always responds with $\perp$ in Case 1. That is, for any known invalid encoding h, the response of $\mathcal{G}^{\mathsf{add}}_{N,m_2}(h,\cdot)$ is always $\perp$. For Case 2, as mentioned above, our simulator considers the unknown encoding h to be invalid with the probability close to the one in $\mathcal{G}_{N,m_2}$, and then responds with $\perp$.

Below, we give a full proof of Theorem 5.

Proof. According to the definition of indifferentiability, the adversary has two honest interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1}$ and two adversarial interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_2}, \mathcal{G}^{\mathsf{add}}_{N,m_2}$. Therefore, we need to build an efficient simulator $\mathcal{S}$ in the ideal world that can simulate two adversarial interfaces properly, which means, for any PPT differentiator $\mathcal{D}$, the view of $\mathcal{D}$ in the real game is computationally close to the view in the ideal game. We will go through a sequence of hybrid games where in each game, there exists a system that responds to all of the queries (both honest and adversarial) in a slightly different way and then we build our simulator $\mathcal{S}$ as the system in the last game. Before the description of the games, we first specify some parameters:

- There are four types of queries: the long labeling query $(x, \mathsf{label}; \mathcal{G}_{N,m_2})$, the long addition query $(h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, the short labeling query $(x, \mathsf{label}; \mathcal{G}_{N,m_1})$ and the short addition query $(\tilde{h}_1, \tilde{h}_2, \mathsf{add}; \mathcal{G}_{N,m_1})$, where $x \in \mathbb{Z}_N$, $h_1, h_2 \in \{0,1\}^{m_2}$ and $\tilde{h}, \tilde{h}_1, \tilde{h}_2 \in \{0,1\}^{m_1}$.
- The adversary only makes q queries to the system, where $q = \mathsf{poly}(\lambda)$.
- The oracles used in the real world are a short cryptographic group construction $\tilde{\mathcal{G}}_{N,m_1} := (\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1})$ and a long generic group $\tilde{\mathcal{G}}_{N,m_2} := (\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}, \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2})$.
- In each game, the system's responses are denoted as $\mathcal{G}^{\mathsf{labelr}}_{N,m_2}$, $\mathcal{G}^{\mathsf{addr}}_{N,m_2}$, $\mathcal{G}^{\mathsf{labelr}}_{N,m_1}$ and $\mathcal{G}^{\mathsf{addr}}_{N,m_1}$. For instance, $\mathcal{G}^{\mathsf{labelr}}_{N,m_2}(x)$ denotes the system's response when adversary makes a query $\mathsf{que} := (x, \mathsf{label}; \mathcal{G}_{N,m_2})$.

The hybrid games are as follows.

Game 0. This game is identical to the real game except that the system maintains a tables T_{label} and T_{add}. Specifically, the system responds to every queries by real oracles. For the tables, the system maintains them as follows:

- T_{label}: It is initiated empty and consists of tuples with form of $(x, \tilde{h}, h)$. Once the adversary makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_2})$, which does not exist in T_{label} yet, the system inserts $(x, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x), \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x))$ into T_{label}.
- T_{add}: It is initiated empty and consists of tuples with form of (h_1, h_2, h). Once the adversary makes a query $(h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, which does not exist in T_{add} yet, the system inserts $(h_1, h_2, \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h_1, h_2))$ into T_{add}.
- $T_{\perp}$: It is initiated empty and consists of tuples with form of $(\tilde{h}, h)$.

Note that the tables T_{label}, T_{add} and $T_{\perp}$ are completely hidden to the adversary, and hence the view in real game is identical to the one in Game 0, which refers to

$$\Pr[\text{Real Game}] = \Pr[\text{Game 0}]$$

Game 1. This game is identical to Game 0 except the way of maintaining the tables and responding to the queries. Specifically,
Labeling query. Suppose $\mathsf{que}_k = (x, \mathsf{label}; \mathcal{G}_{N,m_2})$(the k-th querh), then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{label}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_1 = x$, then the system responds to the query with t_3.
- Case 2: If $\mathsf{que}_k \notin T_{\mathsf{label}}$, the system first makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_1})$ with the response $\tilde{h}$, then
  - Case 2.1: If there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_2 = \tilde{h}$, then the system replaces t_1 with x and responds with t_3.

– Case 2.2: Otherwise, the system responds with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x)$ and inserts $(x, \tilde{h}, h)$ into T_{label}.

Addition query. Suppose $\mathsf{que}_k = (h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{add}}$ such that $\{t_1, t_2\} = \{h_1, h_2\}$, then the system responds to the query with t_3.
- Case 2: If $\mathsf{que}_k \in T_{\perp}$, then the system responds to the query with $\perp$.
- Case 3: If there exist two tuples $(t_1, t_2, t_3), (t_1', t_2', t_3') \in T_{\mathsf{label}}$ such that $t_3 = h_1, t_3' = h_2$ and $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t_2') \in T_{\mathsf{label}}$, then the system responds with the corresponding record.
- Case 4: If there exist two tuples $(t_1, t_2, t_3), (t_1', t_2', t_3') \in T_{\mathsf{label}}$ such that $t_3 = h_1, t_3' = h_2$ but $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t_2') \notin T_{\mathsf{label}}$, then the system responds to the query with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h_1, h_2)$, inserts (h_1, h_2, h) into T_{add}, and inserts $(\diamond, \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(t_1 + t_1'), h)$ into T_{label}.
- Case 5: Otherwise, the system responds to the query with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_2}(h_1, h_2)$ and inserts (h_1, h_2, h) into T_{add}.

In Game 1, the system keeps longer tables, and for part of the adversarial queries, the system responds to them only using the tables. Note that, each item stored in tables is consistent with the real oracles, these responses are identical to those by calling real oracles. Moreover, in either games, the honest interfaces always correspond to the real oracles. Hence, in either game, the response of any query is identical, which refers to

$$\Pr[\text{Game 0}] = \Pr[\text{Game 1}]$$

Now, the system needs to answer the rest adversary queries by real oracles. In the following hybrid games, we replace the responses that answered by real oracles with tables and honest interfaces without changing the view significantly.

Game 2. This game is identical to Game 1 except for responding to the labeling queries. Suppose $\mathsf{que}_k = (x, \mathsf{label}; \mathcal{G}_{N,m_2})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{label}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_1 = x$, then same as Game 1.
- Case 2: If $\mathsf{que}_k \notin T_{\mathsf{label}}$, the system first makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_1})$ with the response $\tilde{h}$, then
  - Case 2.1: If there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_2 = \tilde{h}$, then same as Game 1.
  - Case 2.2: Otherwise, the system uniformly samples $b \in \{0, 1\}$ and responds to the query with $h = \tilde{h}||b$. In addition, the system inserts $(x, \tilde{h}, h)$ into T_{label}.

Note that, the only difference between Game 1 and Game 2 occurs in Case 2, where x never appears before que_k. In Game 1, the system responds with $\tilde{\mathcal{G}}_{N,m_2}$; while in Game 2, the system responds with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)||b$, where $b \in \{0, 1\}$ is a random string. According to the construction, the adversary can only learn the first m_1 bits of the long encoding of the previously unknown x. In other word, the last one bit is uniformly random and independent of the adversary's view. Due to the uniqueness of $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)$, h is also not repeated. Hence, the adversary's views of $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x)$ and $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)||b$ are distributed identically, which refers to

$$\Pr[\text{Game 1}] = \Pr[\text{Game 2}]$$

Game 3. This game is identical to Game 2 except for responding to the addition queries. Suppose $\mathsf{que}_k = (h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, then same as Game 2.

- Case 2: If $\mathsf{que}_k \in T_\perp$, then same as Game 2.
- Case 3: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_3 = h_1, t'_3 = h_2$ and $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2) \in T_{\mathsf{label}}$, then same as Game 2.
- Case 4: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_3 = h_1, t'_3 = h_2$ but $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2) \notin T_{\mathsf{label}}$, then the system randomly samples b from $\{0, 1\}$ and responds to the query with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2)||b$.
- Case 5: Otherwise, same as Game 2.

Note that, the only difference between Game 2 and Game 3 occurs in Case 3. In Game 2, the system responds with $\tilde{\mathcal{G}}_{N,m_2}$; while in Game 3, the system responds with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(t_2, t'_2)||b$, where $b \in \{0, 1\}$ is a random bit. According to the construction, the adversary can only learn the first m_1 bits of the long encoding of the previously unknown x. In other word, the last one bit is uniformly random and independent of the adversary's view. Since $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2)$ does not appear, h does not appear either. Hence, the adversary's views of que_k in Game 2 and Game 3 are distributed identically, which refers to

$$\Pr[\text{Game 2}] = \Pr[\text{Game 3}]$$

Game 4. This game is identical to Game 3 except for responding to the addition queries. Suppose $\mathsf{que}_k = (h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, then same as Game 3.
- Case 2: If $\mathsf{que}_k \in T_\perp$, then same as Game 3.
- Case 3: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_3 = h_1, t'_3 = h_2$ and $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2) \in T_{\mathsf{label}}$, then same as Game 3.
- Case 4: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_3 = h_1, t'_3 = h_2$ but $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2) \notin T_{\mathsf{label}}$, then same as Game 3.
- Case 5: For $i \in \{1, 2\}$, if h_i does not exist in T_{label}, the system first tests whether $\tilde{h}_i \leftarrow \mathsf{Trunc}(h_i)$ is a group element in $\tilde{\mathcal{G}}_{N,m_1}$. If not, the system responds with $\perp$; else if $\tilde{h}_i \in T_{\mathsf{label}}$, then the system also responds with $\perp$.
- Case 6: Otherwise, same as Game 3.

Note that, the only difference between Game 3 and Game 4 occurs in Case 5, where the first m_1-bit string of a previously unknown encoding already exist in T_{label}. In Game 3, the system responds with $\tilde{\mathcal{G}}_{N,m_2}$; while in Game 4, the system always responds with $\perp$. Obviously, the bad event is that such unknown encoding, say h_i, is a valid group element. Due to our construction, when $\tilde{h}_i$ is not a group element, h_i is always invalid. The bad event happens only if both $\tilde{h}||0$ and $\tilde{h}||1$ are valid group elements in the generic group $\tilde{\mathcal{G}}_{N,m_2}$, which is bounded as

$$\Pr[\text{Bad}] \leq \frac{2^{(m_2 - m_1)} \cdot N}{2^{m_2}} = \frac{N}{2^{m_1}}$$

Therefore, we have

$$|\Pr[\text{Game 4}] - \Pr[\text{Game 3}]| \leq 2q \cdot \Pr[\text{Bad}]$$

Game 4. This game is identical to Game 4 except for responding to the addition queries. Suppose $\mathsf{que}_k = (h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, then same as Game 4.
- Case 2: If $\mathsf{que}_k \in T_\perp$, then same as Game 4.

- Case 3: If there exist two tuples $(t_1, t_2, t_3), (t_1', t_2', t_3') \in T_{\mathsf{label}}$ such that $t_3 = h_1, t_3' = h_2$ and $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t_2') \in T_{\mathsf{label}}$, then same as Game 4.

- Case 4: If there exist two tuples $(t_1, t_2, t_3), (t_1', t_2', t_3') \in T_{\mathsf{label}}$ such that $t_3 = h_1, t_3' = h_2$ but $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t_2') \notin T_{\mathsf{label}}$, then same as Game 4.

- Case 5: For $i \in \{1, 2\}$, if h_i does not exist in T_{label}, the system first tests whether $\tilde{h}_i \leftarrow \mathsf{Trunc}(h_i)$ is a group element in $\tilde{\mathcal{G}}_{N,m_1}$. If $\tilde{h}_i$ is invalid or $\tilde{h}_i \in T_{\mathsf{label}}$, then same as Game 4.

- Case 6: Otherwise, if h_1 does not exist in T_{label}, the system samples k_1 from the Bernoulli distribution with the probability $\frac{1}{2-|T_\perp[\tilde{h}_1]|}$ for 1, where $\tilde{h}_1 = \mathsf{Trunc}(h_1)$. If $k_1 = 0$, the system inserts $(\tilde{h}_1, h_1)$ into $T_\perp$; otherwise, the system inserts $(\diamond, \tilde{h}_1, h_1)$ into T_{label}. Similarly for h_2. Finally, the system responds to que_k in the same way as in Case 2, Case 3 or Case 4.

Note that, the only difference between Game 3 and Game 4 occurs in Case 6, where h_1 or h_2 never appears, and the first m_1-bit strings of these unknown encodings are short group elements and do not exist in T_{label}.

We observe that, in Game 3, any previously unknown encoding, say h_i, is valid encoding with a probability bounded by $\frac{1}{2-|T_\perp[\tilde{h}_i]|} + \frac{N}{2^{m_1}}$[24]; while in Game 4, the probability is $\frac{1}{2-|T_\perp[\tilde{h}_i]|}$ due to the Bernoulli distribution sample. Therefore, the adversary's view are distributed closely as

$$(\frac{1}{2-|T_\perp[\tilde{h}_i]|} + \frac{N}{2^{m_1}}) - \frac{1}{2-|T_\perp[\tilde{h}_i]|} = \frac{N}{2^{m_1}}.$$

When the system finally responds to que_k in the same way as in Case 2, Case 3 or Case 4, recall that, it introduces no bad event. Therefore, we have

$$|\Pr[\text{Game 4}] - \Pr[\text{Game 3}]| \leq \frac{N}{2^{m_1}}$$

Ideal Game. In Game 4, the queries to the adversarial interfaces are answered by the tables which are maintained by the system and by making queries to honest interface. It is straightforward to show that we can replace $\tilde{\mathcal{G}}_{N,m_1}$ with a generic group $\mathcal{G}_{N,m_1}$, resulting in Ideal Game.

The difference between Game 4 and Ideal Game is that: in Game 4, the system responds to all queries by calling $\tilde{\mathcal{G}}_{N,m_1}$; while in Ideal Game, the system makes queries to $\mathcal{G}_{N,m_1}$. In fact, as $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x) = \mathsf{Trunc}(\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_2}(x))$ in which $\tilde{\mathcal{G}}_{N,m_2}$ is a generic group, all bits of $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}(x)$ are uniformly random. Analogously, if h_1, h_2 are valid group elements, all bits of $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(h_1, h_2)$ are uniformly random. Thus, the distribution of $\tilde{\mathcal{G}}_{N,m_1}$ and $\mathcal{G}_{N,m_1}$ are identical, referring to

$$\Pr[\text{Game 4}] = \Pr[\text{Ideal Game}]$$

In the following, we give the full description of the simulator.

Simulator in the Ideal Game. Let $\mathcal{G}_{N,m_1}$ be a generic group. By definition, the simulator $\mathcal{S}$ has access to the honest interfaces $\mathcal{G}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1}$. And for the adversarial queries, $\mathcal{S}$ works as the system in Game 7. Concretely, $\mathcal{S}$ maintains four tables: the labeling table T_{label}, the addition table T_{add}, and the invalid table $T_\perp$. And $\mathcal{S}$ answers the adversarial queries by the tables and the honest interfaces $\tilde{\mathcal{G}}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1}$.
Labeling query. Suppose $\mathsf{que}_k = (x, \mathsf{label}; \mathcal{G}_{N,m_2})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{label}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_1 = x$, then the simulator responds with t_3.

- Case 2: If $\mathsf{que}_k \notin T_{\mathsf{label}}$, the system first makes a query $(x, \mathsf{label}; \mathcal{G}_{N,m_1})$ with the response $\tilde{h}$, then

[24]The addition $\frac{N}{2^{m_1}}$ is from the probability of both $\tilde{h}_i||0$ and $\tilde{h}_i||1$ are valid group elements.

– Case 2.1: If there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{label}}$ such that $t_2 = \tilde{h}$, then the simulator responds with t_3.
– Case 2.2: Otherwise, the simulator uniformly samples $b \in \{0, 1\}$ and responds to the query with $h = \tilde{h}||b$. In addition, the simulator inserts $(x, \tilde{h}, h)$ into T_{label}.

Addition query. Suppose $\mathsf{que}_k = (h_1, h_2, \mathsf{add}; \mathcal{G}_{N,m_2})$, then

- Case 1: If $\mathsf{que}_k \in T_{\mathsf{add}}$, which means there exists a tuple $(t_1, t_2, t_3) \in T_{\mathsf{add}}$ such that $\{t_1, t_2\} = \{h_1, h_2\}$, then the simulator responds to the query with t_3.
- Case 2: If $\mathsf{que}_k \in T_\perp$, then the simulator responds to the query with $\perp$.
- Case 3: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_3 = h_1, t'_3 = h_2$ and $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2) \in T_{\mathsf{label}}$, then the system responds with the corresponding record.
- Case 4: If there exist two tuples $(t_1, t_2, t_3), (t'_1, t'_2, t'_3) \in T_{\mathsf{label}}$ such that $t_3 = h_1, t'_3 = h_2$ but $\tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2) \notin T_{\mathsf{label}}$, then the system randomly samples b from $\{0, 1\}$ and responds to the query with $h \leftarrow \tilde{\mathcal{G}}^{\mathsf{add}}_{N,m_1}(t_2, t'_2)||b$.
- Case 5: For $i \in \{1, 2\}$, if h_i does not exist in T_{label}, the system first tests whether $\tilde{h}_i \leftarrow \mathsf{Trunc}(h_i)$ is a group element in $\tilde{\mathcal{G}}_{N,m_1}$. If $\tilde{h}_i$ is invalid or $\tilde{h}_i \in T_{\mathsf{label}}$, then the simulator responds with $\perp$.
- Case 6: Otherwise, if h_1 does not exist in T_{label}, the simulator samples k_1 from the Bernoulli distribution with the probability $\frac{1}{2 - |T_\perp[\tilde{h}_1]|}$ for 1, where $\tilde{h}_1 = \mathsf{Trunc}(h_1)$. If $k_1 = 0$, the simulator inserts $(\tilde{h}_1, h_1)$ into $T_\perp$; otherwise, the simulator inserts $(\diamond, \tilde{h}_1, h_1)$ into T_{label}. Similarly for h_2. Finally, the simulator responds to que_k in the same way as in Case 2, Case 3 or Case 4.

We have shown that every pair of adjacent games are indistinguishable. Combining together, we establish the entire proof, referring to

$$\Big|\Pr[\text{Real Game}] - \Pr[\text{Ideal Game}]\Big| \leq \frac{2qN + N}{2^{m_1}}.$$

□

Acknowledgment. Cong Zhang was supported by the National Key Research and Development Program of China (No. 2023YFB3106000). This work was also supported by Ant Group through CCF-Ant Research Fund. Bingsheng Zhang was supported by the National Natural Science Foundation of China (Grant No. 62072401 and No. 62232002) and Input Output (iohk.io). Hong-Sheng Zhou was supported in part by NSF grant CNS-1801470 and a VCU Research Quest grant.

References

[ABD+13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 531–550. Springer, Heidelberg, August 2013.

[Bar20] Elaine Barker. Recommendation for key management: Part 1 – general, 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5.

[BF18] Manuel Barbosa and Pooya Farshim. Indifferentiable authenticated encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 187–220. Springer, Heidelberg, August 2018.

[BKSY11] Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 559–578. Springer, Heidelberg, March 2011.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

[CDMS10] Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 273–289. Springer, Heidelberg, February 2010.

[CHK$^+$16] Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, January 2016.

[CMR$^+$23] Lily Chen, Dustin Moody, Karen Randall, Andrew Regenscheid, and Angela Robinson. Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters, 2023. `https://doi.org/10.6028/NIST.SP.800-186`.

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

[DHH$^+$21] Nico Döttling, Dominik Hartmann, Dennis Hofheinz, Eike Kiltz, Sven Schäge, and Bogdan Ursu. On the impossibility of purely algebraic signatures. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 317–349. Springer, Heidelberg, November 2021.

[DRS09] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for practical applications. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, Heidelberg, April 2009.

[DSSL16] Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of confusion-diffusion networks. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 679–704. Springer, Heidelberg, May 2016.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 82–115. Springer, Heidelberg, November 2017.

[GWL23] Chun Guo, Lei Wang, and Dongdai Lin. Impossibility of indifferentiable iterated blockciphers from 3 or less primitive calls. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 408–439. Springer, Heidelberg, April 2023.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 89–98, New York, NY, USA, 2011. Association for Computing Machinery.

[HMQS23] Mohammad Hajiabadi, Mohammad Mahmoody, Wei Qi, and Sara Sarfaraz. Lower bounds on assumptions behind registration-based encryption. In Guy Rothblum and Hoeteck Wee, editors, *Theory of Cryptography*, pages 306–334, Cham, 2023. Springer Nature Switzerland.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

[MM11] Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 597–614. Springer, Heidelberg, March 2011.

[MMN16] Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. On the impossibility of virtual black-box obfuscation in idealized models. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 18–48. Springer, Heidelberg, January 2016.

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

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

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

[PH78] Stephen Pohlig and Martin 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.

[PRV12] Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012. https://eprint.iacr.org/2012/653.

[RS08] Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 433–450. Springer, Heidelberg, August 2008.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

[RSS20] Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 155–180. Springer, Heidelberg, May 2020.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2004.

[SGS20] Gili Schul-Ganz and Gil Segev. Accumulators in (and beyond) generic groups: Non-trivial batch verification requires interaction. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 77–107. Springer, Heidelberg, November 2020.

[SGS21] Gili Schul-Ganz and Gil Segev. Generic-group identity-based encryption: A tight impossibility result. In *Information Theoretic Cryptography*, 2021.

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

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Heidelberg, August 2022.

[ZZ18] Mark Zhandry and Cong Zhang. Impossibility of order-revealing encryption in idealized models. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 129–158. Springer, Heidelberg, November 2018.

[ZZ20] Mark Zhandry and Cong Zhang. Indifferentiability for public key cryptosystems. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 63–93. Springer, Heidelberg, August 2020.

[ZZ23] Cong Zhang and Mark Zhandry. The relationship between idealized models under computationally bounded adversaries. In *ASIACRYPT 2023*, 2023.