Attention is still what you need: Another Round of Exploring Shoup's GGM

Taiyu Wang[1,2], Cong Zhang[1,2(✉)], Hong-Sheng Zhou[3], Xin Wang[4], Pengfei Chen[1,2], Wenli Wang[1,2], Kui Ren[1,2], and Chun Chen[1,2]

[1] The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China
congresearch@zju.edu.cn
[2] Hangzhou High-Tech Zone (Binjiang) Blockchain and Data Security Research Institute, Hangzhou, China
[3] Virginia Commonwealth University, Richmond, USA
[4] Digital Technologies, Ant Group, Hangzhou, China

Abstract. The generic group model (GGM) is fundamental for evaluating the feasibility and limitations of group-based cryptosystems. Two prominent versions of the GGM exist in the literature: Shoup's GGM and Maurer's GGM. Zhandry (CRYPTO 2022) points out inherent limitations in Maurer's GGM by demonstrating that several textbook cryptographic primitives, which are provably secure in Shoup's GGM, cannot be proven secure in Maurer's model.

In this work, we further investigate Shoup's GGM and identify novel limitations that have been previously overlooked. Specifically, to prevent generic algorithms from generating valid group elements without querying the oracle, the model typically employs sufficiently large encoding lengths. This leads to sparse encodings, a setting referred to as the sparse generic group model (sparse GGM). We emphasize that this sparseness introduces several constraints:

- **Groups with AE and Black-Box Separation:** Shoup's GGM is typically instantiated with elliptic curve groups, which admit admissible encodings (AE)—functions mapping from $\mathbb{Z}_p$ to elliptic curve points. We establish a black-box separation, showing that the sparse GGM fails to capture cryptographic groups that are both (1) computational Diffie-Hellman (CDH) secure and (2) compatible with admissible encodings.
- **Comparison with EC-GGM:** We examine the relationship between the sparse GGM and the Elliptic Curve Generic Group Model (EC-GGM) introduced by Groth and Shoup (EUROCRYPT 2022), which inherently yields CDH-secure groups with admissible encodings. Within the framework of indifferentiability, we prove that EC-GGM is strictly stronger than sparse GGM.
- **Dense Groups and Black-Box Separation:** We revisit groups with dense encodings and establish a black-box separation between CDH-secure dense groups and the sparse GGM.

The work was mainly supported by National Key Research and Development Program of China, Grant No. 2023YFB3106000.

G. Hanaoka and B.-Y. Yang (Eds.): ASIACRYPT 2025, LNCS 16248, pp. 239–271, 2026.
https://doi.org/10.1007/978-981-95-5113-2_8

– **Extension to Bilinear Settings:** Our results naturally extend to the sparse Generic Bilinear Group Model (GBM), demonstrating that the aforementioned constraints still hold.

In conclusion, our findings indicate that both feasibility and impossibility results in Shoup's GGM should be reinterpreted in a fine-grained manner, encouraging further exploration of cryptographic constructions and black-box separations in EC-GGM or dense GGM.

1 Introduction

Since the seminal work of Diffie and Hellman [DH76], group-based cryptosystems have become a cornerstone of modern cryptography, enabling secure key exchange, public key encryption, digital signatures, and many more. Building on this foundation, various types of cryptographic groups have been proposed, including multiplicative groups [DH76,ElG85], elliptic curve (EC) groups [Mil86,Kob87], and pairing-friendly groups [FR94,BGOS07], all of which play crucial roles in constructing cryptographic schemes and protocols.

In practical group-based cryptosystems, the **length** of group encodings plays a crucial role in efficiency, particularly affecting communication complexity. At equivalent security levels, groups with more compact encodings are typically preferred for implementation. NIST SP 800-186 [CMR+23] lists several recommended curves with 128-bit security, including Curve 25519, an elliptic curve defined over a 255-bit prime field. With standard point compression, a group element on Curve 25519 can be encoded in 256 bits (255 bits plus 1 for sign). Because EC groups generally offer more compact encodings at the same security level, they are widely favored for practical cryptographic constructions.

Importantly, EC groups always have **admissible encodings**; see [BF03, Ica09,BCI+10,LPS23]. Admissible encodings are broadly defined as efficiently computable functions that map from $\mathbb{Z}_p$[1] to group elements, satisfying two key properties: regularity (having constant preimage sizes) and preimage-computability (enabling efficient computation of all preimages for a given element in the range). These encodings enable the *oblivious sampling* of group elements without revealing their discrete logarithms, a critical feature for many group-based cryptosystems [BF03,BBB+18,MR19].

However, a fundamental limitation arises when attempting to prove the security of group-based cryptosystems—the inability to establish the unconditional hardness of the underlying computational assumptions (e.g., discrete logarithm, computational or decisional Diffie-Hellman) for any specific group. Over the past few decades, researchers have explored various approaches to establishing lower bounds on computational hardness, with the most prevalent method being the generic group model (GGM), where algorithms are limited to performing only generic group operations.

[1] Here, p denotes the prime modulus used in elliptic curve groups, where the curve is typically defined by the equation $y^2 = x^3 + Ax + B \bmod p$.

Roughly speaking, there are two widely accepted versions of the generic group model: Maurer's GGM [Mau05] and Shoup's GGM [Sho97]. Maurer's GGM is modeled as a stateful system where algorithms make queries by referencing two group elements encountered during the computation (e.g., the 5th and 9th elements). In contrast, Shoup's GGM is modeled as a random injection from the additive group $\mathbb{Z}_N$ to *sufficiently long* strings, where algorithms specify queries by providing two previously encountered strings from the computation.

Although both formulations appear in the literature, Maurer's GGM is generally viewed as more restrictive. In particular, digital-signature schemes are impossible in Maurer's GGM [DHH+21], and foundational constructions such as the Blum-Micali pseudorandom generator and the Goldreich-Goldwasser-Micali pseudorandom function cannot be realized in this model [Zha22]. For these reasons, we focus exclusively on Shoup's GGM.

In Shoup's GGM, the length of group encodings also plays a significant role. Numerous applications (e.g., [Gro16, Zha22, HMQS23, LZ24]) within this model rely on the requirement that algorithms cannot produce valid group elements without querying the oracle. To uphold this property, the generic group model must employ sparse encodings, making the sparse GGM a critical component of Shoup's framework. However, sparse GGM does not support oblivious sampling, which implies that ***admissible encodings are effectively absent in sparse GGM!***

Attention is still needed! A potential gap arises between sparse variant of Shoup's GGM and practical cryptographic groups, such as elliptic curve groups or other dense groups. Therefore, we plan to conduct further exploration of Shoup's GGM, leading to the following research question:

> *Can sparse GGM effectively and accurately model elliptic curve groups or other dense cryptographic groups?*

1.1 Our Results

In this work, we answer the question above negatively. Specifically, let G be a cryptographic group such that: (1) computational Diffie-Hellman (CDH) is assumed to hold with respect to $\mathbb{G}$; (2) the group encodings are dense or the group is associated with *nice* admissible encodings. Then $\mathbb{G}$ does not exist in the sparse GGM unconditionally.

In order to understand our results better, we start with a brief explanation of several important concepts including (1) sparse generic group models (GGMs), (2) dense groups, and (3) groups with nice admissible encodings. Let λ be the security parameter,

– *Sparse and dense GGMs.* We say GGM $\mathcal{G}$ is *sparse* if the GGM represents a random injection from $\mathbb{Z}_N$ to S, where N is a λ-bit prime, $S = \{0,1\}^m$, and the difference $m - \lambda$ satisfying $m - \lambda \geq \omega(\log \lambda)$. Conversely, we say the GGM $\mathcal{G}$ is *dense* if the difference $m - \lambda \leq \Theta(\log \lambda)$.

– *Sparse and dense group encodings.* Similarly, we say a cryptographic group G of order N is *sparse* if the length of its group encodings, denoted by m, satisfies $m - \lambda \geq \omega(\log \lambda)$, and we say the group is *dense* if $m - \lambda \leq \Theta(\log \lambda)$.
– *Groups with nice admissible encodings.* We define a group $\mathbb{G}$ of order N to have a *nice admissible encoding* with respect to a constant ϱ and a polynomial $\mathsf{poly}_{\mathsf{AE}}$ if there exists an efficiently computable function $\mathsf{AE} : \mathbb{Z}_p \to \mathbb{G}$ satisfying the following conditions: (1) the preimage of any group element under AE is efficiently computable; (2) the encoding is ϱ-regular, meaning the size of any preimage set is at most ϱ; (3) the domain size p is sufficiently large, i.e., $\frac{p}{N} \geq \mathsf{poly}_{\mathsf{AE}}$. As shown in [Ica09], all elliptic curve groups satisfy these conditions with $\varrho = 4$, and are therefore considered groups with nice admissible encodings.

1.1.1 Impossibility Results in Sparse GGM/GBM

Impossibility of Groups with Admissible Encodings in a Sparse GGM. As previously discussed, the sparse GGM does not provide an admissible encoding. We argue that this limitation is an inherent drawback of the sparse GGM and proceed to establish a black-box separation between CDH-secure groups with admissible encodings and the sparse GGM.

Theorem 1.1 (Informal). *For any constant ϱ, CDH-secure groups with admissible encodings with respect to ϱ do not exist in the sparse generic group model.*

Impossibility of Dense Groups in a Sparse GGM. We now turn our attention to the dense groups. Intuitively, the encodings in sparse GGM cannot be compressed in a generic manner and we demonstrate that this limitation represents another inherent drawback of the sparse GGM and proceed to establish a black-box separation between CDH-secure dense groups and the sparse GGM.

Theorem 1.2 (Informal). *CDH-secure dense groups do not exist in the sparse generic group model.*

Remark 1.1. *In [JZW+24], Ji et al. investigate the relationship between CDH-secure groups with varying lengths of group encodings and demonstrate that shorter CDH-secure groups are separated from longer GGMs. However, we highlight that their analysis heavily depends on the assumption that both the groups and the GGM share the same security parameter—a condition we believe is not fully justified. In contrast, our results establish the black-box separation between dense groups and sparse GGM, resolving the open problem posed in [JZW+24].*

Impossibility of Groups with Admissible Encodings or Dense Groups in a Sparse GBM. We further argue that the inherent difficulty in constructing groups with admissible encodings or dense groups persists as long as the idealized model remains sparse, even when the model is extended to the generic bilinear group model (GBM). The GBM $\mathcal{B}$ models two generic groups, i.e., the source generic group and the target generic group, and we say $\mathcal{B}$ is sparse if both the

source generic group and the target generic group are sparse. We then establish black-box separations between CDH-secure groups with admissible encodings (CDH-secure dense groups) and the sparse GBM.

Theorem 1.3 (Informal). *For any constant ϱ, CDH-secure groups with admissible encodings with respect to ϱ does not exist in the sparse generic bilinear group model.*

Theorem 1.4 (Informal). *CDH-secure dense groups do not exist in the sparse generic bilinear group model.*

1.1.2 Exploring the Relationship Between EC-GGM and Sparse GGM

Groth and Shoup [GS22] introduce a variant of the GGM, known as the *elliptic curve generic group model* (EC-GGM). Let E be the set of all points on an elliptic curve group of order N. The EC-GGM models a random injection from $\mathbb{Z}_N$ to E, while preserving certain structural properties inherent to elliptic curves—for instance, the preservation of which points share the same x-coordinate. It follows that CDH-secure groups with nice admissible encodings exist relative to the EC-GGM.

Theorem 1.5 (Informal). *CDH-secure groups with nice admissible encodings exist in the elliptic curve generic group model.*

Relationship Between EC-GGM and Sparse GGM. We next study the relationship between the EC-GGM and the sparse GGM in the framework of indifferentiability. Following Zhang and Zhandry's analysis [ZZ23], we explore the relationship against computationally bounded adversary and prove that:

Theorem 1.6 (Informal). *In the framework of indifferentiability, EC-GGM is strictly stronger than sparse GGM.*

Remark 1.2. *In [JZW+24], Ji et al. also prove that, within the framework of indifferentiability, the dense GGM is strictly stronger than the sparse GGM. However, we stress that their analysis also highly depends on the assumption that both the dense GGM and the sparse GGM share the same security parameter, which we believe is not fully justified. In contrast, our separation between the dense GGM and the sparse GGM is unconditional.*

Theorem 1.7 (Informal). *CDH-secure dense groups exist in the dense GGM.*

1.2 Interpretation

Shoup's GGM serves as the foundation for the group-based cryptosystems. Our findings show that extreme caution must be taken when proving security or establishing black-box separations within Shoup's GGM. We next elaborate it from two perspectives.

Instantiating the GGM with Elliptic Curve Groups. In the generic group model, algorithms are required to be generic. However, not all algorithms for group-based assumptions, such as the discrete logarithm problem, adhere to this requirement—for example, the index calculus attack against $\mathbb{Z}_N^*$ [Adl79]. Fortunately, in the case of elliptic curve groups, the only known attacks are generic in nature. As a result, the generic group model is typically instantiated using elliptic curve groups. However, to the best of our knowledge, in many practical cryptographic constructions (e.g., the zk-SNARKs in [Gro16]), the generic group model is often treated as a sparse GGM, where adversaries are restricted from obtaining valid group elements without making explicit queries. Furthermore, as shown in [Ica09], *all* elliptic curve groups fall into the category of groups with nice admissible encodings.

Our results establish a black-box separation between CDH-secure groups with admissible encodings and the sparse GGM, highlighting a potential and previously overlooked gap when instantiating the GGM with elliptic curve groups.

Black-Box Separations within the GGM. The generic group model is used to demonstrate the limitations of certain group-based cryptosystems. Notably, most known separations [Zha22, HMQS23] are established within sparse GGM. Our findings highlight important limitations of the black-box impossibility of sparse GGM, as it excludes both CDH-secure groups with nice admissible encodings and CDH-secure dense groups. Consequently, the relativizing separation between these groups and identity-based encryption (IBE) remains unresolved. Since many practical groups used fall under the category of groups with nice admissible encodings, our results motivate the further study of the complexity of such groups.

Furthermore, our findings indicate that, when examined in a fine-grained manner, the relationship between the GGM and the GBM must be reinterpreted. Specifically, Zhang and Zhandry [ZZ23] demonstrate that the GBM is strictly stronger than the GGM. In contrast, our results reveal that EC-GGM (or dense GGM) and sparse GBM are, in fact, incomparable.

In conclusion, our results encourage further investigation into both the construction of practical schemes and the establishment of black-box separations within the EC-GGM (or dense GGM). Additionally, our findings offer a deeper understanding of the complexities in GGMs and GBMs.

1.3 Technical Overview

We now provide an overview of the core intuition and novelty behind our techniques.

1.3.1 Impossibility of Groups with AE in Sparse GGM.

To demonstrate the impossibility of a cryptographic primitive $\mathcal{P}$ within an idealized model $\mathcal{O}$, we typically employ the black-box separation methodology. Specifically, for any instantiation $\Pi^{\mathcal{O}}$ of $\mathcal{P}$, we construct a computationally unbounded adversary $\mathcal{A}$ such that (1) $\mathcal{A}$ breaks the security of $\Pi^{\mathcal{O}}$; and (2) $\mathcal{A}$ is

query-efficient. The seminal result of Impagliazzo and Rudich [IR89], tells that the key agreement primitive cannot exist in the random oracle model (ROM). Specifically, for any protocol in this model, there exists an adversary capable of winning in a key recovery attack (KRA), which breaks the fundamental security requirement for key agreement. Returning to our context, it is important to note that CDH-secure groups with admissible encodings inherently imply KRA-secure key agreement (e.g., the Diffie-Hellman key exchange protocol). Therefore, if we could establish that KRA-secure key agreement is impossible in sparse GGM, the analysis would be complete.

For clarity of elaboration, we focus our analysis on a particular case of the key agreement primitive, i.e., the non-interactive key exchange (NIKE). We begin by briefly reviewing the key concepts behind the impossibility of KRA-secure NIKE in the random oracle model [IR89,BKSY11], and then proceed to integrate those insights into our analysis.

KRA-Secure NIKE vs. ROM. Let $\Pi^{\mathcal{H}} := (\mathsf{KGen}^{\mathcal{H}}, \mathsf{SHK}^{\mathcal{H}})$ be a NIKE scheme in the random oracle model $\mathcal{H}$, where (1) $\Pi^{\mathcal{H}}$ achieves *perfect* correctness, and (2) both KGen and SHK make at most q queries. Consider Alice and Bob as two honest parties, each associated with public keys pk_1 and pk_2, respectively. We then construct an adversary $\mathcal{A}$ that, given pk_1 and pk_2, outputs the valid shared key with a good probability. Specifically, the adversary $\mathcal{A}$ initializes two empty sets $S_{\mathsf{que\text{-}res}}$ and S_{key}, and then proceeds through $4q+1$ iterations of the following phases:

- *Simulation.* Simulate a proper view for Alice. Specifically, this view contains a set of query/response pairs $\tilde{S}_{\mathcal{A}}$ along with a private key $\widetilde{\mathsf{sk}}_1$ such that: (1) $\tilde{S}_{\mathcal{A}}$ is consistent with $S_{\mathsf{que\text{-}res}}$[2]; and (2) this view induces the correct public key for Alice: $\mathsf{KGen}^{\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}}(\widetilde{\mathsf{sk}}_1) = \mathsf{pk}_1$. Next, compute the shared key based on the simulated view, i.e., $\mathsf{shk} = \mathsf{SHK}^{\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}}(\widetilde{\mathsf{sk}}_1, \mathsf{pk}_2)$, and insert shk into the set S_{key}.
- *Update.* Update all queries in $\tilde{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$ by accessing the random oracle $\mathcal{H}$, and add the corresponding query/response pairs into the set $S_{\mathsf{que\text{-}res}}$.

Finally, $\mathcal{A}$ outputs the majority of the shared keys from the set S_{key}.

Next, we outline the key intuition behind why $\mathcal{A}$ is likely to win in the KRA-secure game with a good probability. Let S_{Bob} denote the set of query/response pairs made by Bob during the real execution of the key exchange protocol. For any iteration, there are two cases:

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

[2] Since $S_{\mathsf{que\text{-}res}}$ is only a subset of $\mathcal{H}$, it is possible that $\tilde{S}_{\mathcal{A}}$ may be inconsistent with the random oracle $\mathcal{H}$.

Observe that case 1 can occur in at most $2q$ iterations, as $|S_{\mathsf{Bob}}| \leq 2q$. Once this case takes place, the update phase will absorb at least one pair $(\mathsf{que}_B, \mathsf{res}_B) \in S_{\mathsf{Bob}}$ into $S_{\mathsf{que\text{-}res}}$. For case 2, we note that when it occurs, there exists an alternate oracle $\tilde{\mathcal{H}}$ that remains consistent with both $\tilde{S}_{\mathcal{A}}$ and S_{Bob}. Due to perfect correctness, the shared key computed during that iteration is guaranteed to be valid. Furthermore, since case 2 occurs in at least $2q+1$ iterations, this ensures that the majority of keys in S_{key} are valid.

However, in the case of GGM, the attack fails immediately, as the GGM itself implies the KRA-secure NIKE. Next, we explain the reasoning behind the failure of the attack.

Why the Attack Fails in GGM? In contrast to the ROM, the GGM features two types of queries: labeling queries (e.g., $x, \mathcal{G}(x)$) and addition queries (e.g., $\mathcal{G}(x), \mathcal{G}(y), \mathcal{G}(x+y)$). Hence, to guarantee the validity of the shared key, we should define S_{Bob} to include all group encodings present in the queries (both labeling and addition) along with their corresponding discrete logarithms (which Bob may not know). Consequently, for any iteration, there are *three* cases:

- Case 1 (**Bad**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \tilde{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_{\mathsf{Bob}}$ s.t. $\mathsf{que}_A = \mathsf{que}_B$ but $\mathsf{res}_A \neq \mathsf{res}_B$.
- Case 2 (**Bad**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \tilde{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_{\mathsf{Bob}}$ s.t. $\mathsf{que}_A \neq \mathsf{que}_B$ but $\mathsf{res}_A = \mathsf{res}_B$.
- Case 3 (**Good**): $\forall(\mathsf{que}_A, \mathsf{res}_A) \in \tilde{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_{\mathsf{Bob}}$, we have that if $\mathsf{que}_A = \mathsf{que}_B$ then $\mathsf{res}_A = \mathsf{res}_B$, and vice versa.

Observe that case 1 and case 3 can be handled similarly as above. However, case 2 may always occur, meaning it is impossible to find a GGM instance that is consistent with both $\tilde{S}_{\mathcal{A}}$ and S_{Bob}. This implies the aforementioned attack fails.

Upon further exploration, we observe that the occurrence of case 2 indicates that the adversary is able to generate a valid group element without knowing its discrete logarithm, i.e., without making labeling queries. Therefore, to advance the analysis, it is necessary to handle the NIKE and GGM with a more fine-grained approach, ensuring that no query-efficient adversary can generate a valid group element without knowing its discrete logarithm. Specifically, the analysis aims to identify a hard problem related to the fine-grained NIKE and GGM, and prove that if an adversary, given the public keys pk_1 and pk_2, succeeds in outputting a valid group element without making queries, then the hard problem does not hold anymore.

Solution in [JZW+24] and Its Limitations. Recently, Ji et.al. [JZW+24] demonstrated that CDH-secure cryptographic groups with shorter group descriptions cannot exist in the generic group model with longer encodings. Specifically, they establish a *somewhat* black-box separation between KRA-secure NIKE schemes associated with shorter public keys and the GGMs with longer group encodings[3]. The key idea in their work is that, given the public keys pk_1 and

[3] The formal primitive can be trivially constructed via shorter CDH-secure groups.

pk_2, the extracted group element str can fall into one of two cases: (1) str is related to both pk_1 and pk_2; or (2) str is related to only one of the public keys but independent of the other. In the former case, Ji et.al. explain that str is typically a frequent query, which can be easily handled by repeatedly running the key generation algorithm on sufficiently many random inputs. In the latter case, where str is only relevant to only one public key (e.g., pk_1), they observe that pk_1 is the sole carrier of information about str. However, since the length of pk_1 is significantly shorter than that of str, it lacks the necessary capacity to encode all the information required for the recovery of str. Consequently, no query-efficient algorithm can extract str with a non-negligible probability.

Furthermore, the hard problem identified in [JZW+24] can be interpreted as follows: given any public key (which is shorter), no query-efficient algorithm can extract a valid group element (which is longer) except with negligible probability.

However, the hard problem in [JZW+24] faces an inherent limitation. Specifically, it holds only if the length of pk_1 is significantly shorter than that of str. This condition is guaranteed when both the KRA-secure NIKE and the GGM *share the same security parameter*. However, if NIKE and the GGM are associated with different security parameters, this condition may no longer hold. Exploring deeper, we observe that the reason for this limitation lies in the fact that "shorter" and "longer" are relative terms with respect to the security parameter. A public key considered "shorter" under a large security parameter may still carry enough information to extract a "longer" GGM encoding when the security parameter is small. Therefore, we argue that the hard problem in [JZW+24] is based on an unjustified assumption, and their analysis *deviates from the conventional black-box separation*[4].

Our Techniques. To establish a black-box separation between groups with nice admissible encodings and sparse GGMs, it is essential to identify a new and *unconditional* hard problem. We leverage the sparseness property of the GGM and define the hard problem as follows: for any query-efficient algorithm with access to the sparse GGM, which only receives the security parameter as input, it cannot output a new and valid group element except with negligible probability[5]. By saying that a group element str is new, we mean that: (1) str has been used as an input in some addition queries made by the algorithm, and (2) str has not appeared as the output of any prior queries. As discussed in [Zha22], the hardness of this problem holds unconditionally. Next we elaborate on the high-level strategy for intergrating this hard problem into the black-box separation.

Roughly speaking, we show that if the adversary $\mathcal{A}$, given inputs pk_1 and pk_2, is able to generate a new group element with a good probability, then we

[4] The conventional black-box separation framework establishes impossibility results unconditionally, whereas [JZW+24] demonstrates impossibility only under certain unjustified conditions.

[5] The probability is taken over the sampling of the GGM instances and the internal randomness of the algorithm.

can construct an extractor $\mathcal{E}$ which, given only the security parameter as input, can also output a new group element with a good probability.

The first technical challenge of our analysis is determining how the extractor $\mathcal{E}$ generates the two public keys. A natural approach would be:

Step 1: $\mathcal{E}$ randomly selects two private keys (sk_1 and sk_2), computes the corresponding public keys ($\mathsf{pk}_i = \mathsf{KGen}^{\mathcal{G}}(\mathsf{sk}_i)$), and runs $\mathcal{A}^{\mathcal{G}}(\mathsf{pk}_1, \mathsf{pk}_2)$.
Step 2: Whenever $\mathcal{A}$ issues an addition query using a new group element (denoted as str) as input, $\mathcal{E}$ outputs str.

Unfortunately, this approach fails immediately. Given the fact that the GGM is sparse, any new group element str generated by $\mathcal{A}$ should be related to either pk_1 or pk_2. More precisely, str is likely to appear during the execution of $\mathsf{KGen}^{\mathcal{G}}(\mathsf{sk}_1)$ or $\mathsf{KGen}^{\mathcal{G}}(\mathsf{sk}_2)$ with high probability. As a result, if following the aforementioned approach, then the extractor would fail to output a new group element, even if $\mathcal{A}$ does so.

To overcome this challenge, we must adopt an alternative method for generating the public keys, leveraging admissible encodings as part of the solution. Specifically, we configure the NIKE as a KRA-secure NIKE with nice admissible encodings, meaning there exists an admissible encoding that maps from $\mathbb{Z}_p$ to the public keys, where p is sufficiently large. Clearly, CDH-secure groups with nice admissible encodings implies such a NIKE. We then construct the extractor $\mathcal{E}$ as follows:

Step 1: $\mathcal{E}$ randomly selects two seeds (seed_1 and seed_2), computes the corresponding public keys via admissible encodings ($\mathsf{pk}_i = \mathsf{AE}^{\mathcal{G}}(\mathsf{seed}_i)$), and runs $\mathcal{A}^{\mathcal{G}}(\mathsf{pk}_1, \mathsf{pk}_2)$.
Step 2: Whenever $\mathcal{A}$ issues an addition query using a new group element (denoted as str) as input, $\mathcal{E}$ outputs str.

At this point, the second technical challenge emerges. Specifically, let S_{AE} denote the set of the query/response pairs that arise during the execution of $\mathsf{pk}_1 = \mathsf{AE}^{\mathcal{G}}(\mathsf{seed}_1)$ and $\mathsf{pk}_2 = \mathsf{AE}^{\mathcal{G}}(\mathsf{seed}_2)$. Note that if the new group element generated by the adversary $\mathcal{A}$ consistently falls within the set S_{AE}, the extractor above will still fail to output a new group element.

To address this challenge, we refine the adversary's strategy. Specifically, we introduce a preprocessing phase for the adversary, define as follows:

- *Preprocessing.* Given the inputs pk_1 and pk_2, the adversary $\mathcal{A}$ runs the inverse of the admissible encodings, computing $\mathsf{seed}_i = \mathsf{inv\text{-}AE}^{\mathcal{G}}(\mathsf{pk}_i)$. Subsequently, $\mathcal{A}$ recalculates $\mathsf{pk}_i = \mathsf{AE}^{\mathcal{G}}(\mathsf{seed}_i)$ and collects all associated query/response pairs into the set $S_{\mathsf{que\text{-}res}}$.

The adversary then proceeds as follows: before entering the iteration phase, it first executes the preprocessing phase.

Next, we explain why the preprocessing phase is effective. It is important to note that, following the preprocessing phase, the adversary $\mathcal{A}$ has already

gathered all the query/response pairs contained in S_{AE}. As a result, whenever $\mathcal{A}$ generates a new group element, it will also be new to $\mathcal{E}$.

The above outline is not comprehensive; for detailed technical explanations, please refer to Sect. 3.

The key idea of our analysis is that valid public keys can be generated through two distinct methods. In the context of NIKE with dense encodings, there are likewise two methods for generating public keys, one of which even does not require making any queries. Thus, our analysis can be naturally extended to establish a black-box separation between CDH-secure dense groups and sparse GGM. Furthermore, the hard problem above remains valid within the sparse generic bilinear group model (GBM), allowing all impossibility results to naturally extend to the sparse GBM as well.

1.3.2 EC-GGM vs. Sparse GGM

To demonstrate that the elliptic curve generic group model (EC-GGM) is strictly stronger than the sparse generic group model (sparse GGM), we frame our goal within framework of indifferentiability. In particular, we establish that EC-GGM (denoted as $\mathsf{ec}\text{-}\mathcal{G}$) statistically implies sparse GGM (denoted as $\mathcal{G}$); while the sparse GGM does not computationally imply EC-GGM.

EC-GGM Statistically Implies Sparse GGM. We begin by explaining how $\mathsf{ec}\text{-}\mathcal{G}$ implies $\mathcal{G}$ against statistical adversaries. Let $\mathsf{ec}\text{-}\mathcal{G} = (\mathsf{ec}\text{-}\mathcal{G}^{\mathsf{L}}, \mathsf{ec}\text{-}\mathcal{G}^{\mathsf{A}})$ denote an EC-GGM corresponding to an elliptic curve defined over $\mathbb{Z}_p$, where each group element is represented as $(u, b) \in \mathbb{Z}_p \times \{0, 1\}$. According to the Hasse bound, the encoding of EC-GGM is inherently dense. To extend the encoding length to m bits for the sparse GGM, we introduce an additional oracle, the random permutation Perm over $\{0,1\}^m$, along with its inverse, $\mathsf{Perm}^{\mathsf{Inv}}$. The labeling function is then defined as follows:

$$\mathsf{L}^{\mathsf{ec}\text{-}\mathcal{G}}(x) := \mathsf{Perm}\left(\mathsf{ec}\text{-}\mathcal{G}^{\mathsf{L}}(x) || 0 \cdots 0\right).$$

The addition algorithm is constructed by applying the inverse oracle $\mathsf{Perm}^{\mathsf{Inv}}$. To demonstrate indifferentiability, we need to construct a simulator $\mathcal{S}$ that, with access to $\mathcal{G}$, can accurately simulate both $\mathsf{ec}\text{-}\mathcal{G}$ and $(\mathsf{Perm}, \mathsf{Perm}^{\mathsf{Inv}})$.

To simulate $\mathsf{ec}\text{-}\mathcal{G}$, the simulator maintains a tabel to record the correspondence between the elliptic curve points $P := (u, b)$ and $\mathsf{str} := \mathcal{G}(x)$. An important property of the EC-GGM is that for any $(u, b) \leftarrow \mathsf{ec}\text{-}\mathcal{G}^{\mathsf{L}}(x)$ and $(u', b') \leftarrow \mathsf{ec}\text{-}\mathcal{G}^{\mathsf{L}}(-x)$, it holds that $u = u'$ and $b = 1 - b'$. Consequently, whenever the simulator records the tuple (P, str), where $P := (u, b)$ and $\mathsf{str} := \mathcal{G}(x)$, it additionally records $(-P, \mathcal{G}(-x))$ with $-P := (u, 1 - b)$. Notably, $\mathcal{G}(-x)$ can be obtained by making additional queries to $\mathcal{G}$ on str, even without knowledge of x.

To simulate Perm, the simulator $\mathcal{S}$ first checks whether the query corresponds to an elliptic curve point and whether there exists a corresponding $\mathcal{G}(x)$. If both conditions are satisfied, $\mathcal{S}$ just responds with $\mathcal{G}$. If not, $\mathcal{S}$ responds with $\mathcal{G}(x)$ for a randomly sampled $x \in \mathbb{Z}_N$ or with a randomly chosen invalid string str,

depending on whether the query corresponds to a valid point. The simulation of $\mathsf{Perm}^{\mathsf{Inv}}$ follows a similar strategy.

Remark 1.3. *Careful readers might argue that, within the elliptic curve generic group model, adversaries could perform certain non-generic operations, such as oblivious sampling. As a result, EC-GGM extends the adversaries' capabilities, necessitating a reinterpretation of the hardness of certain security assumptions (e.g., DLOG or CDH).*

Fortunately, due to statistical indifferentiability, the hardness of the CDH assumption remains intact. Specifically, we demonstrate that if a statistical yet query-efficient adversary exists that can break CDH in EC-GGM, we can construct a differentiator capable of breaking indifferentiability. More precisely, the differentiator acts as the challenger in the CDH game, interacts with the adversary, and distinguishes between the real and ideal worlds by observing whether the adversary successfully wins the CDH game.

Next, we demonstrate that even in the context of computationally bounded adversaries, it is not possible to construct an indifferentiable EC-GGM within a sparse GGM.

Sparse GGM Does Not Computationally Imply EC-GGM. Suppose we have a proposed construction of an elliptic curve generic group model derived from a sparse GGM, denoted as $\Pi^{\mathcal{G}} := (\mathsf{L}^{\mathcal{G}}, \mathsf{A}^{\mathcal{G}})$. How can we demonstrate that a computationally bounded adversary is able to differentiate $\Pi^{\mathcal{G}}$ from $\mathsf{ec}\text{-}\mathcal{G}$? A common approach is to identify a security game that remains secure in $\mathsf{ec}\text{-}\mathcal{G}$ but can be easily broken in any construction of $\Pi^{\mathcal{G}}$.

Following the strategy outlined in [ZZ23], we define the security game as *discrete logarithm identification* (DLI), a variant of the discrete logarithm problem. Intuitively, the DLI game involves the following: given $\mathsf{str} := \mathsf{L}(x)$, construct a probabilistic, efficient, and query-free circuit C such that $C(x)$ accepts with high probability, while $C(x')$ overwhelmingly rejects for all $x' \neq x$.

Next, we briefly recall the techniques from [ZZ23], where Zhang and Zhandry establish a separation between GGM and ROM against computationally bounded adversaries. Clearly, DLI is secure in GGM. To establish the separation, Zhang and Zhandry show that DLI can be easily broken in any group constructed from the ROM. In a nutshell, the adversary constructs the circuit C as follows: $C(\cdot)$ functions identically to $\mathsf{L}^{\mathcal{H}}(\cdot)$, but without access to $\mathcal{H}$, and accepts the input if the reconstructed result matches $\mathsf{L}^{\mathcal{H}}(x)$ (hardwired in C). The key idea in [ZZ23] is to *anticipate* the queries that C will make to the random oracle model. This enables the adversary to make all sensitive queries in advance and hardcode the corresponding query/response pairs into C, thereby creating an oracle-free circuit. Zhang and Zhandry demonstrate that, with high probability, all sensitive queries can be collected as follows:

1. randomly sample $r, r_1, \ldots, r_n$[6], execute $\mathsf{L}^{\mathcal{H}}(r)$, $\mathsf{L}^{\mathcal{H}}(-r)$, $\mathsf{L}^{\mathcal{H}}(r_1), \ldots, \mathsf{L}^{\mathcal{H}}(r_n)$, and collect all the queries into $S_{\mathsf{que\text{-}res}}$;

[6] Here n is a sufficiently large integer.

2. execute $\mathsf{L}^{\mathcal{H}}(x-r) \leftarrow \mathsf{A}^{\mathcal{H}}(\mathsf{L}^{\mathcal{H}}(x), \mathsf{L}^{\mathcal{H}}(-r))$ and $\mathsf{A}^{\mathcal{H}}(\mathsf{L}^{\mathcal{H}}(x-r), \mathsf{L}^{\mathcal{H}}(r))$, and collect all the queries into $S_{\mathsf{que\text{-}res}}$.

Next, we outline our approach for the incorporating aforementioned technique into the analysis within sparse GGM. Specifically, given an input $\mathsf{L}^{\mathcal{G}}(x)$, the adversary $\mathcal{A}$ collects all the queries as described above, hardwires both the query/response pairs and $\mathsf{L}^{\mathcal{G}}(x)$ into the circuit C, which operates identically to $\mathsf{L}^{\mathcal{G}}(\cdot)$, and then outputs $C(\cdot)$. We then analyze the probability of $\mathcal{A}$ wins. Let Q_x be the set of all query/response pairs generated during the execution of $\mathsf{L}^{\mathcal{G}}(x)$[7], for each pair $(\mathsf{que}, \mathsf{res}) \in Q_x$, there are four possible cases:

- Case 1: The label $\mathsf{L}^{\mathcal{G}}(x)$ does not depend on res at all;
- Case 2: The label $\mathsf{L}^{\mathcal{G}}(x)$ depends on res, but res does not appear in the response when computing $\mathsf{A}^{\mathcal{G}}(\mathsf{L}^{\mathcal{G}}(x-r), \mathsf{L}^{\mathcal{G}}(r))$;
- Case 3: The label $\mathsf{L}^{\mathcal{G}}(x)$ depends on res, which is obtained by making "labeling" query to $\mathcal{G}$ when computing $\mathsf{A}^{\mathcal{G}}(\mathsf{L}^{\mathcal{G}}(x-r), \mathsf{L}^{\mathcal{G}}(r))$;
- Case 4: The label $\mathsf{L}^{\mathcal{G}}(x)$ depends on res, which is obtained by making "addition" query to $\mathcal{G}$ when computing $\mathsf{A}^{\mathcal{G}}(\mathsf{L}^{\mathcal{G}}(x-r), \mathsf{L}^{\mathcal{G}}(r))$;

Next, we present the analysis for handling each of the four cases.

For $(\mathsf{que}, \mathsf{res})$ in case 1 (same as in [ZZ23]), referred to as a **non-sensitive query**, since $\mathsf{L}^{\mathcal{G}}(x)$ does not depend on res, we can replace the response to que with a uniformly sampled string without affecting the final outcome.

For $(\mathsf{que}, \mathsf{res})$ in case 2 (same as in [ZZ23]), referred to as a **sensitive but frequent query**, since the computation of $\mathsf{A}^{\mathcal{G}}(\mathsf{L}^{\mathcal{G}}(x-r), \mathsf{L}^{\mathcal{G}}(r))$ does not obtain res by making queries, res must be extracted from the inputs, i.e., $\mathsf{L}^{\mathcal{G}}(x-r)$ and/or $\mathsf{L}^{\mathcal{G}}(r)$. This indicates that, with high probability, $(\mathsf{que}, \mathsf{res}) \in Q_x \cap (Q_{x-r} \cup Q_r)$. Moreover, since x, $x-r$ and r are pairwise independent, which means that $Q_x \cap Q_{x-r}$ and $Q_x \cap Q_r$ only contains "frequent" queries. Therefore, this query/response tuple can be collected by running $\mathsf{L}^{\mathcal{G}}(\cdot)$ on sufficiently many random inputs.

For $(\mathsf{que}, \mathsf{res})$ in case 3 (same as in [ZZ23]), referred to as a **sensitive labeling query**, we can collect all labeling queries made during the computation of $\mathsf{A}^{\mathcal{G}}(\mathsf{L}^{\mathcal{G}}(x-r), \mathsf{L}^{\mathcal{G}}(r))$ directly.

For $(\mathsf{que}, \mathsf{res})$ in case 4 (different from [ZZ23]), referred to as a **sensitive addition query**, where $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ consists of two valid group elements. Although res appears in this query, collecting this kind of query is not usually useful for our purpose. Specifically, when executing $L^{\mathcal{G}}(x)$, the algorithm may only issue labeling queries at points $(x_1, \ldots, x_q)$, whereas $S_{\mathsf{que\text{-}res}}$ might only contain query/response pairs in the form of addition, such as $(\mathcal{G}(y_i), \mathcal{G}(z_i), \mathcal{G}(y_i + z_i))$, without explicitly knowing y_i or z_i. Consequently, during the reconstruction of $\mathsf{L}^{\mathcal{G}}(x)$, the algorithm may issue labeling queries at certain points, but C is unable to identify which tuple corresponds to the correct response, resulting in the reconstruction failure.

[7] Without loss of generality, we assume that $\mathsf{L}^{\mathcal{G}}(\cdot)$ only makes labeling queries.

To overcome this technical challenge, we once again leverage the sparseness property of the GGM. Note that the occurrence of the query in case 4 indicates that the adversary is able to generate a new and valid group element without knowing its discrete logarithm. Thus, by applying the same analysis as outlined in Sect. 1.3.1, we can construct an extractor that, given only the security parameter as input, is able to produce a new and valid group element whenever the query in case 4 occurs. This demonstrates that the query in case 4 occurs with negligible probability.

2 Preliminaries

Notations. In this paper, let $\lambda \in \mathbb{Z}$ denote the security parameter. We use $x||y$ to represent the concatenation of the strings x and y. For a finite set S, we denote a random sample s from S according to the uniform distribution as $s \xleftarrow{\$} S$, and the size of S as $|S|$. For a probabilistic algorithm Alg, we overload the notation and write $y \xleftarrow{\$} \mathsf{Alg}(I)$ to denote that the variable y is obtained by running Alg on input I, where I may be a tuple $I = (I_1, ..., I_n)$. If Alg is deterministic, we use the notation "$\leftarrow$" instead of "$\xleftarrow{\$}$".

A positive function $\mathsf{negl}(\cdot)$ is said to be negligible if, for all positive polynomial $\mathsf{poly}(\cdot)$, there exists a constant $\lambda_0 > 0$ such that for all $\lambda > \lambda_0$, it holds that $\mathsf{negl}(\lambda) < 1/\mathsf{poly}(\lambda)$. We say that a function $\rho(\cdot)$ is noticeable in λ if its inverse, $1/\rho(\lambda)$, is polynomial in λ.

2.1 Groups with Admissible Encodings

In this work, we treat groups with admissible encodings as a cryptographic primitive. We first recall the formal definition of a primitive given by [RTV04].

Definition 2.1 (Cryptographic Primitive [RTV04]). *A primitive $\mathcal{P}$ is a pair $\langle \mathcal{F}, \mathcal{R} \rangle$, where $\mathcal{F}$ is a set of functions $f : \{0,1\}^* \mapsto \{0,1\}^*$ specifying correctness property, and $\mathcal{R}$ is a relation on pairs $\langle f, \mathcal{A} \rangle$, with $f \in \mathcal{F}$ and $\mathcal{A}$ an adversarial machine, specifying security property.*

- **Efficient implementation.** *A function f is said to* implement *$\mathcal{P}$ or be an implementation of $\mathcal{P}$ if $f \in \mathcal{F}$. An* efficient implementation *of $\mathcal{P}$ is an implementation of $\mathcal{P}$ that is computable in polynomial time.*
- **Secure implementation.** *An adversarial machine $\mathcal{A}$ is said to* $\mathcal{P}$-break *$f \in \mathcal{F}$ if $\langle f, \mathcal{A} \rangle \in \mathcal{R}$. A* secure implementation *of $\mathcal{P}$ is an implementation of $\mathcal{P}$ such that no* PPT *adversarial machine can $\mathcal{P}$-break f.*

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

To ensure correctness, we consider cryptographic groups that support an efficiently computable encoding from $\mathbb{Z}_p$ (for some integer p) into the group. Various formulations of such admissible encodings appear in the literature [BF03, BCI+10, LPS23]; in this work, we adopt the definition from [LPS23].

Definition 2.2 (Admissible Encodings). *A function* $\mathsf{AE} : S \mapsto T$ *between two finite sets is an admissible encoding with respect to* ϱ *if it satisfies the following conditions:*

Computable*: The function* AE *is computable in polynomial time.*
Sampleable*: Given any element* t *in the image of* AE*, one can efficiently compute its full preimage* $\mathsf{inv\text{-}AE}(t)$*.*
ϱ***-regular***: For any* $t \in T$*, the size of the preimage* $|\mathsf{inv\text{-}AE}(t)|$ *is at most* ϱ*, where* ϱ *is a samll constant.*

Remark 2.1. *The definition of admissible encodings in [BCI+10] differs slightly from that in Definition 2.2. Specifically, it requires that for uniformly distributed inputs* $s \in S$*, the output distribution* $\mathsf{AE}(s)$ *is statistically close to uniform over* T*. However, most known admissible encodings—such as Icart's encoding [Ica09, FT10]—do not produce uniformly distributed outputs. In this work, we adopt the definition from [LPS23], and note that the formulation in [BCI+10] can be seen as a special case of Definition 2.2 with* $\varrho = 1$*.*

When working with admissible encodings for cryptographic groups, the target set T is typically identified with the set of group elements. A key application of admissible encodings is *oblivious sampling*, which requires that the encoding's image covers a noticeable fraction of the set T. This, in turn, requires the domain to be sufficiently large. Specifically, we say an admissible encoding AE for a group $\mathbb{G}$ is *nice* if: (1) AE maps from $\mathbb{Z}_p$ to $\mathbb{G}$, and (2) the ratio $\frac{p}{|\mathbb{G}|}$ is not small.

As shown in [Ica09], all elliptic curve groups admit nice admissible encodings. We now formalize the notion of CDH-secure groups equipped with nice admissible encodings.

Definition 2.3. (CDH-Secure Group with Nice Admissible Encodings). *The CDH-secure group with nice admissible encoding with respect to a constant* ϱ *and a polynomial* $\mathsf{poly}_{\mathsf{AE}}$*, denoted* $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{CDH}}$*, is defined as a pair* $\langle \mathsf{ae}\text{-}\mathcal{F}^{\mathsf{CDH}}, \mathsf{ae}\text{-}\mathcal{R}^{\mathsf{CDH}} \rangle$*, satisfying the following conditions:*

1. *The set* $\mathsf{ae}\text{-}\mathcal{F}^{\mathsf{CDH}}$ *consists of* group-generation functions f *that, on input a security parameter,* f *outputs the description of a finite cyclic group along with an additional function. Specifically, we write* $(\mathbb{G}, g, N, \mathsf{AE}) \leftarrow f(1^\lambda)$*, where* $\mathbb{G}$ *is a cyclic group of* λ*-bit prime order* N*,* g *is a generator of* $\mathbb{G}$*, and* $\mathsf{AE} : \mathbb{Z}_p \mapsto \mathbb{G}$ *is a* ϱ*-regular admissible encoding, with* $p \geq N/\mathsf{poly}_{\mathsf{AE}}(\lambda)$*.*
2. *The pair* $\langle f, \mathcal{A} \rangle$ *belongs to* $\mathsf{ae}\text{-}\mathcal{R}^{\mathsf{CDH}}$ *for a function* $f \in \mathsf{ae}\text{-}\mathcal{F}^{\mathsf{CDH}}$ *and a* PPT *adversarial machine* $\mathcal{A}$ *if there exists a polynomial* $\mathsf{poly}_{\mathcal{A}}(\cdot)$ *such that*

$$\Pr[\mathcal{A}(\mathbb{G}, g, N, \mathsf{AE}, g^{x_1}, g^{x_2}) = g^{x_1 x_2}] > 1/\mathsf{poly}_{\mathcal{A}}(\lambda)$$

for infinitely many values of λ*, where* $(\mathbb{G}, g, N, \mathsf{AE}) \leftarrow f(1^\lambda)$ *and* x_1, x_2 *are chosen uniformly from* $\mathbb{Z}_N$*.*

We say that a CDH-secure group with nice admissible encoding $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{CDH}}$ *exists if there is a function* $f \in \mathsf{ae}\text{-}\mathcal{F}^{\mathsf{CDH}}$ *such that no* PPT *adversarial machine* $\mathcal{A}$ *satisfies* $\langle f, \mathcal{A} \rangle \in \mathsf{ae}\text{-}\mathcal{R}^{\mathsf{CDH}}$*.*

2.2 Dense Groups

We also consider cryptographic groups with dense encodings and formalize this primitive following the definition style in [RTV04].

Definition 2.4. (CDH-Secure Dense Group). *The CDH-secure dense group, denoted* $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{CDH}}$*, is defined as a pair* $\langle \mathsf{d}\text{-}\mathcal{F}^{\mathsf{CDH}}, \mathsf{d}\text{-}\mathcal{R}^{\mathsf{CDH}} \rangle$*, satisfying the following conditions:*

1. *The set* $\mathsf{d}\text{-}\mathcal{F}^{\mathsf{CDH}}$ *consists of* group-generation functions f *that, on input of a security parameter* λ*, output the description of a finite cyclic group with a dense encoding. Specifically, we write* $(\mathbb{G}, g, N, m) \leftarrow f(1^\lambda)$*, where* $\mathbb{G}$ *is a cyclic group of* λ*-bit prime order* N*,* g *is a generator of* $\mathbb{G}$*, and* m *is the length of the group encoding. Here,* m *satisfies* $m - \log N \leq \Theta(\log \lambda)$*, meaning that each group element in* $\mathbb{G}$ *can be represented as an* m*-bit string.*
2. *The pair* $\langle f, \mathcal{A} \rangle$ *belongs to* $\mathsf{d}\text{-}\mathcal{R}^{\mathsf{CDH}}$ *for a function* $f \in \mathsf{d}\text{-}\mathcal{F}^{\mathsf{CDH}}$ *and a* PPT *adversarial machine* $\mathcal{A}$ *if there exists a polynomial* $\mathsf{poly}_{\mathcal{A}}(\cdot)$ *such that*

$$\Pr[\mathcal{A}(\mathbb{G}, g, N, m, g^{x_1}, g^{x_2}) = g^{x_1 x_2}] > 1/\mathsf{poly}_{\mathcal{A}}(\lambda)$$

for infinitely many values of λ*, where* $(\mathbb{G}, g, N, m) \leftarrow f(1^\lambda)$ *and* x_1, x_2 *are chosen uniformly from* $\mathbb{Z}_N$*.*

We say that a CDH-secure dense group $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{CDH}}$ *exists if there is a function* $f \in \mathsf{d}\text{-}\mathcal{F}^{\mathsf{CDH}}$ *such that no* PPT *adversarial machine* $\mathcal{A}$ *satisfies* $\langle f, \mathcal{A} \rangle \in \mathsf{d}\text{-}\mathcal{R}^{\mathsf{CDH}}$*.*

2.3 Idealized Models

In this subsection, we introduce the idealized models discussed in this paper, including the Generic Group Model (GGM) [Sho97], the Generic Bilinear Group Model (GBM) [BB04], and the Elliptic Curve Generic Group Model (EC-GGM) [GS22]. In each model, all entities, including the adversary and the challenger, have access to the corresponding oracle. Below, we specify the behavior of the oracle in each idealized model.

Definition 2.5. (Generic Group Model [Sho97]). *For a given security parameter* $\lambda \in \mathbb{N}$*, let* $\mathcal{I}_{\mathbb{Z}_N, S}$ *denote the set of all injections from* $\mathbb{Z}_N$ *to* S*, where* N *is a* λ*-bit prime number and* $S = \{0,1\}^m$*. The generic group model* $\mathcal{G}_{N,m}$ *is an idealized model that samples a random injection* σ *from* $\mathcal{I}_{\mathbb{Z}_N, S}$ *and provides two oracles,* $\mathcal{G}^{\mathsf{label}}_{N,m}$ *and* $\mathcal{G}^{\mathsf{add}}_{N,m}$*. Concretely,*

- *The labeling oracle* $\mathcal{G}^{\mathsf{label}}_{N,m}$ *takes as input* $x \in \mathbb{Z}_N$ *and returns* $\sigma(x)$*;*
- *The addition oracle* $\mathcal{G}^{\mathsf{add}}_{N,m}$ *takes as input strings* $\mathsf{str}, \mathsf{str}' \in S$ *and behaves as follows: if there exist* $x, x' \in \mathbb{Z}_N$ *such that* $\sigma(x) = \mathsf{str}$ *and* $\sigma(x') = \mathsf{str}'$*, the oracle returns* $\sigma(x + x')$*; otherwise, it returns* $\perp$*.*

In this work, we further say that the generic group model $\mathcal{G}_{N,m}$ is *dense* if $m - \log N \leq \Theta(\log \lambda)$. Otherwise, we refer to it as *sparse*.

Definition 2.6. (Generic Bilinear Group Model [BB04]). *For a given security parameter $\lambda \in \mathbb{N}$, let $\mathcal{I}_{\mathbb{Z}_N,S_i}$ denote the set of all injections from $\mathbb{Z}_N$ to S_i, where N is a λ-bit prime number and $S_i = \{0,1\}^{m_i}$ for $i \in \{1,2,\mathsf{T}\}$. The generic bilinear group model $\mathcal{B}$ is an idealized model that samples random injections σ_1, σ_2, and σ_T from $\mathcal{I}_{\mathbb{Z}_N,S_i}$, respectively, and provides seven oracles: $\mathcal{B}^{1\text{-label}}$, $\mathcal{B}^{1\text{-add}}$, $\mathcal{B}^{2\text{-label}}$, $\mathcal{B}^{2\text{-add}}$, $\mathcal{B}^{\mathsf{T}\text{-label}}$, $\mathcal{B}^{\mathsf{T}\text{-add}}$ and $\mathcal{B}^{\mathsf{map}}$. Concretely,*

- *The labeling oracle $\mathcal{B}^{i\text{-label}}$, for $i \in \{1,2,\mathsf{T}\}$, takes as input $x \in \mathbb{Z}_N$ and returns $\sigma_i(x)$;*
- *The addition oracle $\mathcal{B}^{i\text{-add}}$, for $i \in \{1,2,\mathsf{T}\}$, takes as input strings $\mathsf{str}, \mathsf{str}' \in S_i$, and behaves as follows: if there exist $x, x' \in \mathbb{Z}_N$ such that $\sigma_i(x) = \mathsf{str}$ and $\sigma_i(x') = \mathsf{str}'$, the oracle returns $\sigma_\mathsf{T}(x + x')$; otherwise, it returns $\perp$.*
- *The bilinear map oracle $\mathcal{B}^{\mathsf{map}}$ takes as input strings $\mathsf{str}_1 \in S_1$ and $\mathsf{str}_2 \in S_2$, and behaves as follows: if there exist $x_1, x_2 \in \mathbb{Z}_N$ such that $\sigma_1(x_1) = \mathsf{str}_1$ and $\sigma_2(x_2) = \mathsf{str}_2$, the oracle returns $\sigma_\mathsf{T}(x_1 \cdot x_2)$; otherwise, it returns $\perp$.*

In this work, we consider only symmetric generic bilinear group models, meaning that $\mathcal{B}^{1\text{-label}} = \mathcal{B}^{2\text{-label}}$ and $\mathcal{B}^{1\text{-add}} = \mathcal{B}^{2\text{-add}}$. We make the functions N, m_S and m_T explicit[8], and refer to the model as

$$\mathcal{B}_{N,m_\mathsf{S},m_\mathsf{T}} = (\mathcal{B}^{\mathsf{S}\text{-label}}_{N,m_\mathsf{S},m_\mathsf{T}}, \mathcal{B}^{\mathsf{S}\text{-add}}_{N,m_\mathsf{S},m_\mathsf{T}}, \mathcal{B}^{\mathsf{T}\text{-label}}_{N,m_\mathsf{S},m_\mathsf{T}}, \mathcal{B}^{\mathsf{T}\text{-add}}_{N,m_\mathsf{S},m_\mathsf{T}}, \mathcal{B}^{\mathsf{map}}_{N,m_\mathsf{S},m_\mathsf{T}}).$$

We say that the model $\mathcal{B}_{N,m_\mathsf{S},m_\mathsf{T}}$ is sparse if both $m_\mathsf{S} - \log N \geq \omega(\log \lambda)$ and $m_\mathsf{T} - \log N \geq \omega(\log \lambda)$.

Definition 2.7. (Elliptic Curve Generic Group Model [GS22]). *Let $\lambda \in \mathbb{N}$ be the security parameter. Let E be the set of all points on the elliptic curve $y^2 = F(u)$ over $\mathbb{Z}_p$, where $|E| = N$ and N is a λ-bit prime number[9]. For any point $P = (u,b) \in \mathbb{Z}_p \times \{0,1\}$ in E, we denote by $-P$ the point $(u, 1-b) \in E$. Let $\mathcal{I}_{\mathbb{Z}_N,E}$ be the set of all injections σ from $\mathbb{Z}_N$ to E such that:*

1. *$\sigma(0) = \mathcal{O}$, where $\mathcal{O}$ is the point at infinity.*
2. *For all $x \in \mathbb{Z}_N$, it holds that $\sigma(-x) = -\sigma(x)$.*

The elliptic curve generic group model $\mathsf{ec}\text{-}\mathcal{G}_N$ is an idealized model that samples a random injection σ from $\mathcal{I}_{\mathbb{Z}_N,E}$ and provides two oracles, $\mathsf{ec}\text{-}\mathcal{G}_N^{\mathsf{label}}$ and $\mathsf{ec}\text{-}\mathcal{G}_N^{\mathsf{add}}$. Concretely,

- *The labeling oracle $\mathsf{ec}\text{-}\mathcal{G}_N^{\mathsf{label}}$ takes as input $x \in \mathbb{Z}_N$ and returns $\sigma(x)$;*
- *The addition oracle $\mathsf{ec}\text{-}\mathcal{G}_N^{\mathsf{add}}$ takes as input two points $P, P' \in E$, where $\sigma(x) = P$ and $\sigma(x') = P'$, and returns $\sigma(x + x')$.*

[8] GBM is symmetric indicating that $m_1 = m_2$, and we redenote by m_S the encoding length of source group.

[9] For any $u \in \mathbb{Z}_p$ there are at most two points on the curve with u as their x-coordinate, namely, $(u, \pm y)$ for some y. To simplify, point compression is applied, and we denote the point (u,y) by the pair (u,b), where $b = 0$ if y is even and $b = 1$ if y is odd.

2.4 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]. In the following, we proceed to the definition of indifferentiability:

A cryptosystem Σ consists of a set of algorithms. Here, Σ is accessible via two interfaces Σ.hon and Σ.adv, where Σ.hon provides an honest interface through which the system can be accessed by all parties in a black-box manner, and Σ.adv models the adversarial access to Σ.

Definition 2.8 (Indifferentiability [MRH04]). *Let Σ_1 and Σ_2 be two cryptosystems and $\mathcal{S}$ be a simulator. The indifferentiability advantage of a differentiator $\mathcal{D}$ 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. 1. We say Σ_1 is indifferentiable from Σ_2, if there exists an efficient simulator $\mathcal{S}$ such that for any efficient differentiator $\mathcal{D}$, 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 differentiator $\mathcal{D}$, the advantage above is negligible.

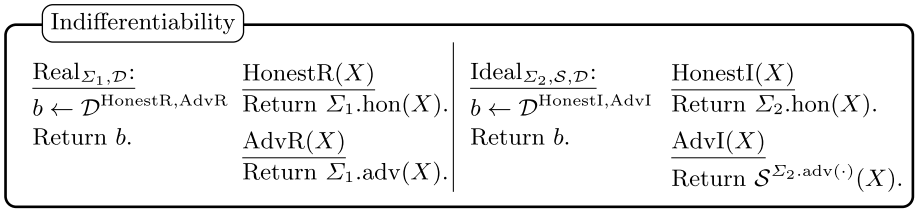

Fig. 1. Indifferentiability of Σ_1 and Σ_2, where $\mathcal{S}$ is the simulator and $\mathcal{D}$ is the adversary.

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 $\mathcal{F}_1$. MRH prove the composition theorem for the

framework of indifferentiability; for simplify, we give a game-based formalization from [RSS11].

Theorem 2.1 (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 differentiator* $\mathcal{D}$ *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 Thm. 2.1 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 2.9 (Computational Indifferentiable Separation [MRH04]). *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 differentiator* $\mathcal{D}_{\Pi,\mathcal{S}}$ *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 Impossibility of Groups with Admissible Encodings

In this section, we elaborate the main constraint of the sparse GGM/GBM. We demonstrate that any the cryptographic group with nice admissible encodings that is CDH-secure, denoted as $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{CDH}}$, cannot exist within the sparse GGM/GBM. Roughly speaking, to establish the black-box separation, the standard approach is to build an adversary that, while computationally unbounded, is query-efficient and capable of breaking the CDH game with respect to any construction of $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{CDH}}$ in the sparse GGM/GBM.

For easier readability, our strategy follows the one in [IR89]. We introduces an intermediary primitive, i.e. non-interactive key exchange (NIKE) with nice admissible encoding that is secure against key-recovery attack (KRA-secure), denoted as $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$, and prove that:

- $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{CDH}}$ implies $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$;
- $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ does not exist in the sparse GGM/GBM.

3.1 Non-interactive Key Exchange with Nice AE

In this section, we present the formal definition of the KRA-secure non-interactive key exchange with nice admissible encoding.

Definition 3.1 (NIKE with Nice AE). *A KRA-secure non-interactive key exchange protocol with nice admissible encoding, denoted* $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$*, is defined as a pair* $\langle \mathsf{ae}\text{-}\mathcal{F}^{\mathsf{NIKE}}, \mathsf{ae}\text{-}\mathcal{R}^{\mathsf{NIKE}} \rangle$*:*

1. *The set* $\mathsf{ae}\text{-}\mathcal{F}^{\mathsf{NIKE}}$ *consists of functions* f*, each associated with a constant* ϱ *and a polynomial* $\mathsf{poly}_{\mathsf{AE}}$ *in* λ*. For a given input security parameter,* f *outputs the description of a non-interactive key exchange protocol along with an additional function. Specifically, we write* $(\mathsf{KGen}, \mathsf{SHK}, \mathsf{AE}) \leftarrow f(1^\lambda)$*, where algorithms are associated with* $\mathcal{SK}, \mathcal{PK}$*, and* $\mathcal{K}$*.*
   - $\mathcal{SK}$*,* $\mathcal{PK}$*, and* $\mathcal{K}$ *are the private-key space, public-key space and shared-key space, respectively, satisfying that* $\mathcal{SK} := \mathbb{Z}_N$*, where* N *is a* λ*-bit integer.*
   - *The* public-key generation function $\mathsf{KGen} : \mathcal{SK} \mapsto \mathcal{PK}$ *is an injection, for generating a public key* $\mathsf{pk} \in \mathcal{PK}$ *from a randomly chosen private key* $\mathsf{sk} \in \mathcal{SK}$*.*
   - *The* shared-key generation function $\mathsf{SHK} : \mathcal{PK} \times \mathcal{SK} \mapsto \mathcal{K} \cup \{\perp\}$ *for generating a shared key* $\mathsf{shk} \in \mathcal{K} \cup \{\perp\}$*, where* $\perp$ *indicates a failed computation.*
   - *The* encoding function $\mathsf{AE} : \mathbb{Z}_p \mapsto \mathcal{PK}$ *is a* ϱ*-regular admissible encoding from* $\mathbb{Z}_p$ *to the codomain of* KGen*, with* $p \geq N/\mathsf{poly}_{\mathsf{AE}}(\lambda)$*.*

   Concretely, for randomly chosen $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$ *and* $\mathsf{sk}' \xleftarrow{\$} \mathcal{SK}$*, compute* $\mathsf{pk} \leftarrow \mathsf{KGen}(\mathsf{sk})$ *and* $\mathsf{pk}' \leftarrow \mathsf{KGen}(\mathsf{sk}')$*. We write* $\mathsf{shk} \leftarrow \mathsf{SHK}(\mathsf{pk}', \mathsf{sk})$ *and* $\mathsf{shk}' \leftarrow \mathsf{SHK}(\mathsf{pk}, \mathsf{sk}')$*. The protocol is required to achieve* perfect correctness*, meaning that:*

$$\Pr\left[\mathsf{shk} = \perp \vee \mathsf{shk}' = \perp \vee \mathsf{shk} \neq \mathsf{shk}'\right] = 0.$$

2. *For a function* $f = (\mathsf{KGen}, \mathsf{SHK}, \mathsf{AE}) \in \mathsf{ae}\text{-}\mathcal{F}^{\mathsf{NIKE}}$ *and a* PPT *(adversarial) machine* $\mathcal{A}$*, we define* $\langle f, \mathcal{A} \rangle \in \mathsf{ae}\text{-}\mathcal{R}^{\mathsf{NIKE}}$ *if* $\mathcal{A}$ *can break the security property of* f *against key-recovery attack (KRA). Specifically, there exists a polynomial* $\mathsf{poly}_{\mathcal{A}}(\cdot)$ *such that:*

$$\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{pk}') = \mathsf{SHK}(\mathsf{pk}', \mathsf{sk}) = \mathsf{SHK}(\mathsf{pk}, \mathsf{sk}') \neq \perp] > 1/\mathsf{poly}_{\mathcal{A}}(\lambda)$$

   for infinitely many values of λ*. Here, for randomly chosen* $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$ *and* $\mathsf{sk}' \xleftarrow{\$} \mathcal{SK}$*, compute* $\mathsf{pk} \leftarrow \mathsf{KGen}(\mathsf{sk})$ *and* $\mathsf{pk}' \leftarrow \mathsf{KGen}(\mathsf{sk}')$*, respectively.*

We say $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ *exists if there is a function* $f \in \mathsf{ae}\text{-}\mathcal{F}^{\mathsf{NIKE}}$ *such that no* PPT *adversarial machine* $\mathcal{A}$ *satisfies* $\langle f, \mathcal{A} \rangle \in \mathsf{ae}\text{-}\mathcal{R}^{\mathsf{NIKE}}$*.*

It is apparent that $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{CDH}}$ implies $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ by defining $\mathsf{KGen}(x) := g^x$ and $\mathsf{SHK}(\mathsf{pk}, \mathsf{sk}') := \mathsf{pk}^{\mathsf{sk}'}$. Therefore, it is sufficient to prove that $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ does not exist in the sparse GGM.

3.2 ae-$\mathcal{P}^{\mathsf{NIKE}}$ Does Not Exist in the Sparse GGM/GBM

In this section, we establish the black-box separation between ae-$\mathcal{P}^{\mathsf{NIKE}}$ and the sparse GGM. Specifically, we construct an adversary that, while computationally unbounded, is query-efficient and capable of breaking the KRA-secure game with a noticeable probability. The corresponding proof for the sparse GBM proceeds analogously and is deferred to the full version of this paper.

Theorem 3.1. *Let $\mathcal{G}_{N,m}$ be a generic group model such that $m - \log N \geq \omega(\log \lambda)$. Let $\Pi^{\mathcal{G}_{N,m}} = (\mathsf{KGen}^{\mathcal{G}_{N,m}}, \mathsf{SHK}^{\mathcal{G}_{N,m}}, \mathsf{AE}^{\mathcal{G}_{N,m}})$ be any non-interactive key exchange protocol with nice admissible encoding parameterized by a constant ϱ and a polynomial $\mathsf{poly}_{\mathsf{AE}}$. Then there exists an adversary $\mathcal{A}$ and two polynomials poly_1 and poly_2 such that:*

- *$\mathcal{A}$ makes poly_1 queries to $\mathcal{G}_{N,m}$;*
- *$\mathcal{A}$ breaks the KRA-secure game with advantage $\frac{1}{\mathsf{poly}_2}$.*

Proof. To establish the proof, we present a formal description of the adversary $\mathcal{A}$, as illustrated in Fig. 2. Here, we specify that any algorithm in $\Pi^{\mathcal{G}_{N,m}}$ makes at most q queries, where q is a polynomial. We then clarify some undefined notions: By $\left\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\right\} \xleftarrow{\text{query}} \mathsf{KGen}^{\mathcal{G}_{N,m}}(x)$, we mean that when running $\mathsf{KGen}^{\mathcal{G}_{N,m}}(x)$, the algorithm makes queries $(\mathsf{que}_1, \ldots, \mathsf{que}_q)$ to the oracle $\mathcal{G}_{N,m}$, and obtains $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$[10].

Trivial to note that the adversary $\mathcal{A}^{\mathcal{G}_{N,m}}$ is query-efficient. We next prove that the adversary $\mathcal{A}^{\mathcal{G}_{N,m}}$ can successfully guess the valid shared key with a noticeable probability. Let $S_{\mathsf{Bob\text{-}L}}$ denote the set of all valid group encodings that appear when running the algorithms $\mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk}_2)$ and $\mathsf{SHK}^{\mathcal{G}_{N,m}}(\mathsf{pk}_1, \mathsf{sk}_2)$; these encodings are either the responses of labeling/addition queries or the valid inputs of the addition queries. Clearly, $|S_{\mathsf{Bob\text{-}L}}| \leq 6q$, since each algorithm makes at most q queries to $\mathcal{G}_{N,m}$. We then define:

$$S_{\mathsf{Bob}} := \left\{(x, \mathsf{str}) | \mathsf{str} \in S_{\mathsf{Bob\text{-}L}},\ \mathcal{G}_{N,m}^{\mathsf{label}}(x) = \mathsf{str}\right\}.$$

Note that in each iteration, if the encoding pairs in $\tilde{S}_{\mathcal{A}}$ are consistent with S_{Bob}, the shared key computed in that iteration would be correct. Specifically, in such a context, there exists another instance of GGM that is consistent with both the simulation view of $\mathcal{A}$ and the real view of user Bob. The perfect correctness of ae-$\mathcal{P}^{\mathsf{NIKE}}$ guarantees that the shared key computed in this iteration must be equal to the true key computed by Bob. However, without knowledge of sk_2, $\tilde{S}_{\mathcal{A}}$ might be inconsistent with S_{Bob} with high probability. In fact, there are three events:

- Event 1: There exist $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}}$ and $(x', \mathsf{str}') \in S_{\mathsf{Bob}}$ such that $x = x'$ but $\mathsf{str} \neq \mathsf{str}'$.

[10] As explained above, we assume that the algorithms $\mathsf{KGen}^{\mathcal{G}_{N,m}}$ and $\mathsf{AE}^{\mathcal{G}_{N,m}}$ only make labeling queries.

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

$\mathcal{A}^{\mathcal{G}_{N,m}}(\mathsf{pk}_1, \mathsf{pk}_2)$:

$S_{\mathsf{key}} \leftarrow \emptyset$; $S_{\mathsf{que\text{-}res}} \leftarrow \emptyset$;
if $\mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_1) = \emptyset$ or $\mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_2) = \emptyset$: return $\perp$;

Preprocessing phase: *//collect queries from admissible encoding*

for every $\mathsf{seed} \in \mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_1) \cup \mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_2)$:
  $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\text{query}} \mathsf{AE}^{\mathcal{G}_{N,m}}(\mathsf{seed})$; *//queries from AE*
  $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}$;

for $i \in \{1, 2\}$: *//queries from AE-inverse*
  $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\text{query}} \mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_i)$;
  for $j = 1$ to q:
    if que_j is a labeling query: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(\mathsf{que}_j, \mathsf{res}_j)\}$;
    if $\mathsf{que}_j = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query:
      if $\exists (x_1, \mathsf{str}_1), (x_2, \mathsf{str}_2) \in S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(x_1 + x_2, \mathsf{res}_j)\}$;

Iteration phase:

for $i = 1$ to $12q + 1$:
  Simulation: search a proper view of Alice
  search a private key $\widetilde{\mathsf{sk}}_1$ and a set of encoding pairs $\tilde{S}_{\mathcal{A}}$ satisfying the following properties:
  1. $\tilde{S}_{\mathcal{A}}$ is consistent with $S_{\mathsf{que\text{-}res}}$ and $|\tilde{S}_{\mathcal{A}}| \leq 12q$;
  2. $\mathsf{pk}_1 \leftarrow \mathsf{KGen}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}_1)$;
  3. $\tilde{S}_{\mathcal{A}}$ is sufficient for the algorithm $\widetilde{\mathsf{shk}}_i \leftarrow \mathsf{SHK}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\mathsf{pk}_2, \widetilde{\mathsf{sk}}_1)$:
     - if $\mathsf{que} = x$ is a labeling query, then x should be covered in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$;
     - if $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query and there are (x_1, str_1), (x_2, str_2) in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, then $(x_1 + x_2, \mathsf{str}_3)$ should also be covered in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$;
     - if $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query but there is no (x_1, str_1) and/or (x_2, str_2) in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, then respond with $\perp$.
  $S_{\mathsf{key}} \leftarrow S_{\mathsf{key}} \cup \{\widetilde{\mathsf{shk}}_i\}$;
  Updating: replace the guessing labeling queries with valid encodings
  for each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup (x, \mathcal{G}^{\mathsf{label}}_{N,m}(x))$;

Final phase: *//output the guessing shared key*

return the majority value in S_{key}.

Fig. 2. The description of $\mathcal{A}$ that breaks the KRA-secure game of $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ in $\mathcal{G}_{N,m}$.

– Event 2: There exist $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}}$ and $(x', \mathsf{str}') \in S_{\mathsf{Bob}}$ such that $x \neq x'$ but $\mathsf{str} = \mathsf{str}'$.
– Event 3: For any $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}}$ and $(x', \mathsf{str}') \in S_{\mathsf{Bob}}$, we have that if $x = x'$ then $\mathsf{str} = \mathsf{str}'$, and vice versa.

We immediately observe that Event 1 occurs at most $6q$ times, since the updating phase would eliminate at least one pair in S_{Bob} with each occurrence. Next we show that, with a noticeable probability, Event 3 would deduce the valid

shared key. According to the adversary, decipted in Fig. 2, we note that the set $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ responds to the labeling queries properly. For the addition queries, say $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$, there are two cases:

- Case 1: $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ covers $(x_1, \mathsf{str}_1), (x_2, \mathsf{str}_2)$, and $(x_1 + x_2, \mathsf{str}_3)$;
- Case 2: either str_1 or str_2 is not collected in $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$.

For the former case, $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ responds to the addition query properly; for the latter case, $\tilde{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ always responds with $\perp$, which means that if both str_1 and str_2 are valid group elements then the response is invalid. Therefore, the only bad case that prevents Event 3 from guessing a valid shared key is that the adversary $\mathcal{A}$ generates valid group elements str without making labeling queries (i.e., without knowing the corresponding discrete logarithm). Besides, we observe that, when Event 2 occurs, the adversary also generates valid group elements without making labeling queries. Hence, if the probability of such a bad case is small, then the adversary $\mathcal{A}^{\mathcal{G}_{N,m}}$ can break the KRA-secure game and output the valid shared key with a good probability.

According to the description of the adversary $\mathcal{A}^{\mathcal{G}_{N,m}}$, we immediately observe that, $\mathcal{A}^{\mathcal{G}_{N,m}}$ aborts if either pk_1 or pk_2 has no preimage with respect to $\mathsf{AE}^{\mathcal{G}_{N,m}}$. To analyze the probability of such a bad event, we define a tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$ is invertible if the following conditions are satisfied:

1. $\mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk}_1)) \neq \emptyset$;
2. $\mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk}_2)) \neq \emptyset$.

Due to the fact that $\mathsf{AE}^{\mathcal{G}_{N,m}}$ is a ϱ-regular admissible encoding from $\mathbb{Z}_p$ to the N valid public keys, there are at least $\left\lceil \frac{p}{\varrho} \right\rceil$ public keys with preimages under $\mathsf{AE}^{\mathcal{G}_{N,m}}$. Each such public key corresponds to a unique private key. Therefore, we have that

$$\begin{aligned}\Pr[(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2) \text{ is invertible}] &\geq \frac{1}{\varrho^2} \cdot \left(\frac{p}{N}\right)^2 \\ &\geq \frac{1}{\varrho^2} \cdot \frac{1}{\mathsf{poly}_{\mathsf{AE}}(\lambda)^2},\end{aligned}$$

where the probability of over the sampling of the instance of the GGM and private keys.

Moreover, we observe that once the randomness of the adversary is fixed, then the algorithm $\mathcal{A}^{\mathcal{G}_{N,m}}(\mathsf{pk}_1, \mathsf{pk}_2)$ is deterministic. Let r be a nonce, we say the adversary performs good with respect to the tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2, r)$, if the adversary $\mathcal{A}^{\mathcal{G}_{N,m}}(\mathsf{pk}_1, \mathsf{pk}_2)$, utilizing r as its source of randomness, is unable to generate a valid group element without knowing the discrete logarithm. Here, pk_i represents the public key generated from the private key sk_i.

Next, we introduce the concept of "good" for the tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$. More formally, we say a tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$ is good if

$$\Pr[\mathcal{A} \text{ performs good with respect to } (\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2, r)] \geq \frac{1}{2},$$

where $\mathsf{pk}_1 \leftarrow \mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk}_1)$ and $\mathsf{pk}_2 \leftarrow \mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk}_2)$, and the probability is over the internal randomness of $\mathcal{A}$. Analogously, we say a tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$ is bad if

$$\Pr[\mathcal{A} \text{ performs good with respect to } (\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2, r)] < \frac{1}{2}.$$

For clarity, we denote these three events in which the tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$ is invertible, good or bad as $\mathsf{Invertible}$, Good and Bad, respectively. Due to the perfect correctness of $\Pi^{\mathcal{G}_{N,m}}$, it is apparent that the probability $\mathcal{A}$ wins the KRA-secure game, conditioned on $\mathsf{Good} \wedge \mathsf{Invertible}$, is $\geq \frac{1}{2}$. Thus, the probability of $\mathcal{A}$ wins can be bounded by

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &\geq \Pr[\mathcal{A}^{\mathcal{G}_{N,m}} \text{ wins} | \mathsf{Good} \wedge \mathsf{Invertible}] \cdot \Pr[\mathsf{Good} \wedge \mathsf{Invertible}] \\ &\geq \frac{1}{2} \cdot \Pr[\mathsf{Good} \wedge \mathsf{Invertible}] = \frac{1}{2} \cdot \Pr[\mathsf{Invertible}] \cdot \Pr[\mathsf{Good} | \mathsf{Invertible}] \\ &\geq \frac{1}{2\varrho^2} \cdot \frac{1}{\mathsf{poly}_{\mathsf{AE}}(\lambda)^2} \cdot \Pr[\mathsf{Good} | \mathsf{Invertible}].\end{aligned}$$

Therefore, it suffices to prove that $\Pr[\mathsf{Good}|\mathsf{Invertible}]$ is noticeable. To finalize the proof, we utilize the sparsity property of the GGM. As discussed in [Zha22], given access to a sparse GGM, any algorithm that receives only the security parameter as input cannot output a valid group element without knowledge of the corresponding discrete logarithm except with negligible probability[11]. For ease of exposition, we refer to this event as the algorithm outputs a new group element. More precisely, we say an algorithm $\mathcal{E}$ outputs a new group element str, if (1) str has been used as an input to some addition queries made by $\mathcal{E}$, and (2) str has not appeared as the output of any previous queries (whether labeling or addition) made by $\mathcal{E}$.

We note that, if $\Pr[\mathsf{Good}|\mathsf{Invertible}]$ is small, then, conditioned on $\mathsf{Invertible}$, the adversary $\mathcal{A}$ is likely to generate a new group element with a good probability. Consequently, our proof strategy proceeds in two steps:

1. Construct an extractor $\mathcal{E}$ that takes only the security parameter as input;
2. Demonstrate that, if $\Pr[\mathsf{Good}|\mathsf{Invertible}]$ is small, then $\mathcal{E}$ outputs a new group element with a good probability.

We now present the formal description of the extractor $\mathcal{E}$, as depicted in Fig. 3. The extractor $\mathcal{E}$ takes only the security parameter as input, randomly selects two seeds (i.e., seed_1 and seed_2), and computes the corresponding public keys (i.e., pk_1 and pk_2) using admissible encodings. Following this, the extractor proceeds in a manner closely resembling the adversary in the KRA-security game, as shown in Fig. 2. Specifically, the extractor proceeds the preprocessing phase and the iteration phase. During the simulation process of each iteration phase, if a new and valid group element appears, either in $\tilde{S}_{\mathcal{A}}$ or as an input of some

[11] We establish the concrete upper bound for this event in the full version of this paper.

Extractor $\mathcal{E}^{\mathcal{G}_{N,m}}(1^\lambda)$

$\mathsf{seed}_1, \mathsf{seed}_2 \xleftarrow{\$} \mathbb{Z}_p$; $\mathsf{pk}_1 \leftarrow \mathsf{AE}^{\mathcal{G}_{N,m}}(\mathsf{seed}_1)$; $\mathsf{pk}_2 \leftarrow \mathsf{AE}^{\mathcal{G}_{N,m}}(\mathsf{seed}_2)$;
$S_{\mathsf{que\text{-}res}} \leftarrow \emptyset$;
Preprocessing phase: *//collect queries from admissible encoding*
for every $\mathsf{seed} \in \mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_1) \cup \mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_2)$:
  $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\text{query}} \mathsf{AE}^{\mathcal{G}_{N,m}}(\mathsf{seed})$; *//queries from AE*
  $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}$;
for $i \in \{1, 2\}$: *//queries from AE-inverse*
  $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\text{query}} \mathsf{inv\text{-}AE}^{\mathcal{G}_{N,m}}(\mathsf{pk}_i)$;
  for $j = 1$ to q:
    if que_j is a labeling query: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(\mathsf{que}_j, \mathsf{res}_j)\}$;
    if $\mathsf{que}_j = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query:
      if $\exists (x_1, \mathsf{str}_1), (x_2, \mathsf{str}_2) \in S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(x_1 + x_2, \mathsf{res}_j)\}$;
      if $\mathsf{str}_1 \notin S_{\mathsf{que\text{-}res}}$ and $\mathcal{G}^{\mathsf{add}}_{N,m}(\mathsf{str}_1, \mathsf{str}_1) \neq \perp$, return str_1;
      if $\mathsf{str}_2 \notin S_{\mathsf{que\text{-}res}}$ and $\mathcal{G}^{\mathsf{add}}_{N,m}(\mathsf{str}_2, \mathsf{str}_2) \neq \perp$, return str_2;
Iteration phase:
for $i = 1$ to $12q + 1$:
  Simulation: search a proper view of Alice
  search a private key $\widetilde{\mathsf{sk}}_1$ and a set of encoding pairs $\tilde{S}_{\mathcal{A}}$ satisfying the following properties:
  1. $\tilde{S}_{\mathcal{A}}$ is consistent with $S_{\mathsf{que\text{-}res}}$ and $|\tilde{S}_{\mathcal{A}}| \leq 12q$;
  2. $\mathsf{pk}_1 \leftarrow \mathsf{KGen}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}_1)$;
  3. $\tilde{S}_{\mathcal{A}}$ is sufficient for the algorithm $\widetilde{\mathsf{shk}}_i \leftarrow \mathsf{SHK}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\mathsf{pk}_2, \widetilde{\mathsf{sk}}_1)$:
     - if $\mathsf{que} = x$ is a labeling query, then x should be covered in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$;
     - if $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query and there are (x_1, str_1), (x_2, str_2) in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, then $(x_1 + x_2, \mathsf{str}_3)$ should also be covered in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$;
     - if $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query but there is no (x_1, str_1) and/or (x_2, str_2) in $S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}$, then respond with $\perp$.
  For each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$: if $\mathcal{G}^{\mathsf{add}}_{N,m}(\mathsf{str}, \mathsf{str}) \neq \perp$, return str;
  Re-run the algorithm $\widetilde{\mathsf{shk}}_i \leftarrow \mathsf{SHK}^{S_{\mathsf{que\text{-}res}} \cup \tilde{S}_{\mathcal{A}}}(\mathsf{pk}_2, \widetilde{\mathsf{sk}}_1)$ and $\mathsf{que} = (\mathsf{str}_1, \mathsf{str}_2)$ is an addition query during the execution:
    if $\mathsf{str}_1 \notin S_{\mathsf{que\text{-}res}}$ and $\mathcal{G}^{\mathsf{add}}_{N,m}(\mathsf{str}_1, \mathsf{str}_1) \neq \perp$, return str_1;
    if $\mathsf{str}_2 \notin S_{\mathsf{que\text{-}res}}$ and $\mathcal{G}^{\mathsf{add}}_{N,m}(\mathsf{str}_2, \mathsf{str}_2) \neq \perp$, return str_2;
  Updating: replace the guessing labeling queries with valid encodings
  for each $(x, \mathsf{str}) \in \tilde{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup (x, \mathcal{G}^{\mathsf{label}}_{N,m}(x))$;
return $\perp$.

Fig. 3. The description of $\mathcal{E}$ w.r.t. $\mathsf{ae}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ that outputs a new group element of $\mathcal{G}_{N,m}$.

addition queries, then $\mathcal{E}$ outputs this element (highlighted in red). Observe that the randomness of $\mathcal{E}$ is composed of three components: seed_1, seed_2, and r, where r represents the randomness used during the iteration phase. Furthermore, once the GGM instance and the two seeds are fixed, the extractor $\mathcal{E}$ proceeds

identically to the adversary $\mathcal{A}$ in the KRA-security game, continuing until $\mathcal{E}$ successfully outputs a new and valid group element.

Next, we analyze the probability of the extractor wins, which occurs when $\mathcal{E}$ outputs a new and valid group element. Let $\mathcal{G}_{N,m}$ denote the GGM to which the extractor $\mathcal{E}$ has access. Let seed_1 and seed_2 denote the seeds sampled by the extractor. We define the seed pair, $(\mathsf{seed}_1, \mathsf{seed}_2)$, to be bad with respect to $\mathcal{G}_{N,m}$, if the tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$, deduced from $(\mathsf{seed}_1, \mathsf{seed}_2)$, is a Bad and $\mathsf{Invertible}$ tuple. More precisely, when we say a tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$ deduced from $(\mathsf{seed}_1, \mathsf{seed}_2)$ with respect to $\mathcal{G}_{N,m}$, we mean that

$$\mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk}_1) = \mathsf{AE}^{\mathcal{G}_{N,m}}(\mathsf{seed}_1) \text{ and } \mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk}_2) = \mathsf{AE}^{\mathcal{G}_{N,m}}(\mathsf{seed}_2).$$

For clarity, we refer to this event as $\mathsf{BadSeed}$ in the following analysis. According to the definition of Bad, we note that if $\mathcal{E}$ fortunately samples a bad seed pair, i.e., $\mathsf{BadSeed}$ occurs, then $\mathcal{E}$ wins with a good probability. Concretely,

$$\Pr[\mathcal{E}^{\mathcal{G}_{N,m}} \text{ wins} | \mathsf{BadSeed}] \geq \frac{1}{2}.$$

Next, we analyze the probability of $\mathsf{BadSeed}$ with respect to any specific GGM instance. To facility the analysis, we introduce several concepts relevant to the generic group model. For any instance of the GGM, denoted as $\mathcal{G}_{N,m}$, we define $Q_{\mathcal{G}_{N,m}}$ as the set of all private keys such that for any private key $\mathsf{sk} \in Q_{\mathcal{G}_{N,m}}$, the corresponding public key (denoted $\mathsf{pk} = \mathsf{KGen}^{\mathcal{G}_{N,m}}(\mathsf{sk})$) has valid preimages under the admissible encodings. Moreover, due to that the admissible encodings is ϱ-regular, it is apparent that

$$\left\lceil \frac{p}{\varrho} \right\rceil \leq |Q_{\mathcal{G}_{N,m}}| \leq N.$$

We then categorize all instances of the GGM into two types: good GGM instances and bad GGM instances. Specifically, let $\mathcal{G}_{N,m}$ be a GGM instance, and $Q_{\mathcal{G}_{N,m}} := \{\mathsf{sk}_1, \ldots, \mathsf{sk}_t\}$. Let (i, j) be a pair where $i, j \in [1, t]$[12]. We classify $\mathcal{G}_{N,m}$ as a bad instance if there are more than $\frac{1}{2\varrho^2} \cdot t^2$ pairs such that each induces a Bad and $\mathsf{Invertible}$ tuple. More precisely, when we say a pair such as (i^*, j^*) induces a Bad and $\mathsf{Invertible}$ tuple, we mean that the tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_{i^*}, \mathsf{sk}_{j^*})$ is bad and invertible. Analogously, we classify $\mathcal{G}_{N,m}$ as a good instance if there are at most $\frac{1}{2\varrho^2} \cdot t^2$ pairs, each of which induces a Bad and $\mathsf{Invertible}$ tuple. Therefore, for any good GGM instance, there are at least $(1 - \frac{1}{2\varrho^2}) \cdot t^2$ tuples that are both Good and $\mathsf{Invertible}$.

Furthermore, for any GGM instance $\mathcal{G}_{N,m}$, if a tuple $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$ is invertible, then there are at least one seed pair, such as $(\mathsf{seed}_1, \mathsf{seed}_2)$, from which $(\mathcal{G}_{N,m}, \mathsf{sk}_1, \mathsf{sk}_2)$ can be deduced from. Consequently, for any fixed GGM instance, we have that,

$$\text{Num of bad and invertible tuples} \leq \text{Num of bad seed pairs}.$$

[12] Here, i and j may be equal.

We are now prepared to establish the lower bound on the probability of the extractor $\mathcal{E}$ wins when accessing $\mathcal{G}_{N,m}$, conditioned on $\mathcal{G}_{N,m}$ being a bad GGM instance. Concretely, let t be the size of $Q_{\mathcal{G}_{N,m}}$, we have that,

$$\begin{aligned}\Pr[\mathcal{E}^{\mathcal{G}_{N,m}} \text{ wins}|\mathcal{G}_{N,m} \text{ is bad}] &\geq \Pr[\mathcal{E}^{\mathcal{G}_{N,m}} \text{ wins}|\mathsf{BadSeed}] \cdot \frac{\text{Num of bad seed pairs}}{\text{Num of all seed pairs}} \\ &\geq \frac{1}{2} \cdot \frac{\text{Num of bad and invertible tuples}}{\text{Num of all seed pairs}} \\ &\geq \frac{1}{2} \cdot \frac{\frac{1}{2\varrho^2} \cdot t^2}{p^2} \\ &\geq \frac{1}{4\varrho^4}.\end{aligned}$$

Next, we analyze the probability of $\mathcal{E}$ wins, where the probability is also considered over the uniform sampling of the GGM instance. Specifically,

$$\begin{aligned}\Pr[\mathcal{E} \text{ wins}] &\geq \Pr[\mathcal{E}^{\mathcal{G}_{N,m}} \text{ wins}|\mathcal{G}_{N,m} \text{ is bad}] \cdot \frac{\text{Num of bad GGM instances}}{\text{Num of all GGM instances}} \\ &\geq \frac{1}{4\varrho^4} \cdot \frac{\text{Num of bad GGM instances}}{\text{Num of all GGM instances}}.\end{aligned}$$

For clarity in exposition, we define $\mathsf{TotalGGM}$ as the set of all GGM instances, BadGGM as the subset of bad GGM instances, and $\mathsf{GoodGGM}$ as the subset of good GGM instances. Moreover, we define

$$\mathsf{BadRatio} := \frac{|\mathsf{BadGGM}|}{|\mathsf{TotalGGM}|}; \mathsf{GoodRatio} := \frac{|\mathsf{GoodGGM}|}{|\mathsf{TotalGGM}|}.$$

Thus, the probability of the extractor $\mathcal{E}$ wins is bounded by

$$\Pr[\mathcal{E} \text{ wins}] \geq \frac{1}{4\varrho^4} \cdot \mathsf{BadRatio}.$$

According to the sparsity property of the GGM, we know that the probability of $\mathcal{E}$ outputs a new group element is negligible, indicating that $\mathsf{BadRatio}$ is negligible. Therefore, it suffices to prove that, if $\Pr[\mathsf{Good}|\mathsf{Invertible}]$ is not noticeable, then $\mathsf{BadRatio}$ is not negligible.

Claim. If $\Pr[\mathsf{Good}|\mathsf{Invertible}] \leq \frac{1}{4\varrho^2} \cdot \frac{1}{\mathsf{poly}_{\mathsf{AE}}(\lambda)^2}$, then $\mathsf{BadRatio} \geq \frac{1}{2}$.

We proceed our analysis by contraposition. Specifically, we prove that if $\mathsf{BadRatio} < \frac{1}{2}$, then $\Pr[\mathsf{Good}|\mathsf{Invertible}] > \frac{1}{4\varrho^2} \cdot \frac{1}{\mathsf{poly}_{\mathsf{AE}}(\lambda)^2}$. By definition, we have that

$$\begin{aligned}\Pr[\mathsf{Good}|\mathsf{Invertible}] &= \frac{\Pr[\mathsf{Good} \wedge \mathsf{Invertible}]}{\Pr[\mathsf{Invertible}]} \\ &= \frac{\Sigma_{\mathcal{G}_{N,m}} \text{Num of } \mathsf{Good} \text{ and } \mathsf{Invertible} \text{ tuples w.r.t. } \mathcal{G}_{\mathrm{N,m}}}{\Sigma_{\mathcal{G}_{N,m}} \text{Num of } \mathsf{Invertible} \text{ tuples w.r.t. } \mathcal{G}_{\mathrm{N,m}}},\end{aligned}$$

where the summation $\Sigma_{\mathcal{G}_{N,m}}$ is taken over all GGM instances. Note that for any GGM instance $\mathcal{G}_{N,m}$, since the size of the private-key space is N, there are at most N^2 relevant tuples associated with $\mathcal{G}_{N,m}$. This implies that

$$\Sigma_{\mathcal{G}_{N,m}} \text{Num of } \mathsf{Invertible} \text{ tuples w.r.t. } \mathcal{G}_{\mathrm{N,m}} \leq |\mathsf{TotalGGM}| \cdot \mathrm{N}^2.$$

Furthermore, by the definition of good GGM instance, for any good GGM instance $\mathcal{G}_{N,m}$, there are at least $(1 - \frac{1}{2\varrho^2}) \cdot t^2$ tuples that are both Good and $\mathsf{Invertible}$ tuples associated with $\mathcal{G}_{N,m}$. Therefore, we know that

$$\Sigma_{\mathcal{G}_{N,m}} \text{Num of } \mathsf{Good} \text{ and } \mathsf{Invertible} \text{ tuples w.r.t. } \mathcal{G}_{\mathrm{N,m}} \geq |\mathsf{GoodGGM}| \cdot (1 - \frac{1}{2\varrho^2}) \cdot \mathrm{t}^2,$$

where $t = |Q_{\mathcal{G}_{N,m}}| \geq \frac{p}{\varrho}$. Thus, we have that

$$\begin{aligned} \Pr[\mathsf{Good}|\mathsf{Invertible}] &\geq \frac{|\mathsf{GoodGGM}| \cdot (1 - \frac{1}{2\varrho^2}) \cdot t^2}{|\mathsf{TotalGGM}| \cdot N^2} \\ &\geq (1 - \mathsf{BadRatio}) \cdot \frac{1}{2\varrho^2} \cdot \frac{1}{\mathsf{poly}_{\mathsf{AE}}(\lambda)^2}. \end{aligned}$$

Therefore, we conclude that, if $\mathsf{BadRatio} < \frac{1}{2}$, then

$$\Pr[\mathsf{Good}|\mathsf{Invertible}] \geq \frac{1}{4\varrho^2} \cdot \frac{1}{\mathsf{poly}_{\mathsf{AE}}(\lambda)^2}.$$

Combining together, we establish the entire proof. □

4 Impossibility of Dense Groups

In this section, we focus on cryptographic groups with dense group encodings, demonstrating that any such group that is CDH-secure, denoted as $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{CDH}}$, cannot exist within the sparse GGM/GBM. Following a similar strategy as in Sect. 3, we introduce an intermediary primitive, i.e. non-interactive key exchange (NIKE) with dense public-key space, which is secure against key-recovery attack (KRA-secure), denoted as $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{NIKE}}$, and prove that:

- $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{CDH}}$ implies $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{NIKE}}$;
- $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ does not exist in the sparse GGM/GBM.

Definition 4.1 (NIKE with Dense Public-Key Space) *A KRA-secure non-interactive key exchange protocol with dense public-key space, denoted as* $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{NIKE}}$*, consists of the pair* $\langle \mathsf{d}\text{-}\mathcal{F}^{\mathsf{NIKE}}, \mathsf{d}\text{-}\mathcal{R}^{\mathsf{NIKE}} \rangle$*:*

1. *The set* $\mathsf{d}\text{-}\mathcal{F}^{\mathsf{NIKE}}$ *consists of functions* f *that, on input of a security parameter, output the description of a non-interactive key exchange protocol. Specifically, we write* $(\mathsf{KGen}, \mathsf{SHK}) \leftarrow f(1^\lambda)$*, where algorithms are associated with* $\mathcal{SK}$*,* $\mathcal{PK}$*, and* $\mathcal{K}$*.*

- $\mathcal{SK}$, $\mathcal{PK}$, *and* $\mathcal{K}$ *are the private-key space, public-key space and shared-key space, respectively, satisfying that* $\mathcal{SK} := \mathbb{Z}_N$, *where* N *is a* λ*-bit integer, and* $\log|\mathcal{PK}| - \log|\mathcal{SK}| \leq \Theta(\log\lambda)$;
- *The* public-key generation function $\mathsf{KGen} : \mathcal{SK} \mapsto \mathcal{PK}$ *is an injection, for generating a public key* $\mathsf{pk} \in \mathcal{PK}$ *from a randomly chosen private key* $\mathsf{sk} \in \mathcal{SK}$.
- *The* shared-key generation function $\mathsf{SHK} : \mathcal{PK} \times \mathcal{SK} \mapsto \mathcal{K} \cup \{\perp\}$ *for generating a shared key* $\mathsf{shk} \in \mathcal{K} \cup \{\perp\}$, *where* $\perp$ *indicates a failed computation.*

Concretely, for randomly chosen $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$ *and* $\mathsf{sk}' \xleftarrow{\$} \mathcal{SK}$ *, compute* $\mathsf{pk} \leftarrow \mathsf{KGen}(\mathsf{sk})$ *and* $\mathsf{pk}' \leftarrow \mathsf{KGen}(\mathsf{sk}')$. *We write* $\mathsf{shk} \leftarrow \mathsf{SHK}(\mathsf{pk}', \mathsf{sk})$ *and* $\mathsf{shk}' \leftarrow \mathsf{SHK}(\mathsf{pk}, \mathsf{sk}')$. *The protocol is required to achieve* perfect correctness, *meaning that:*

$$\Pr\left[\mathsf{shk} = \perp \vee \mathsf{shk}' = \perp \vee \mathsf{shk} \neq \mathsf{shk}'\right] = 0.$$

2. *For a function* $f = (\mathsf{KGen}, \mathsf{SHK}) \in \mathsf{d}\text{-}\mathcal{F}^{\mathsf{NIKE}}$ *and a* PPT *(adversarial) machine* $\mathcal{A}$, *we define* $\langle f, \mathcal{A}\rangle \in \mathsf{d}\text{-}\mathcal{R}^{\mathsf{NIKE}}$ *if* $\mathcal{A}$ *can break the security property of* f *against key-recovery attack (KRA).*

We say $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ *exists if there is a function* $f \in \mathsf{d}\text{-}\mathcal{F}^{\mathsf{NIKE}}$ *such that no* PPT *adversarial machine* $\mathcal{A}$ *satisfies* $\langle f, \mathcal{A}\rangle \in \mathsf{d}\text{-}\mathcal{R}^{\mathsf{NIKE}}$.

It is apparent that $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{CDH}}$ implies $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ by defining $\mathsf{KGen}(x) := g^x$ and $\mathsf{SHK}(\mathsf{pk}, \mathsf{sk}') := \mathsf{pk}^{\mathsf{sk}'}$. Therefore, it is sufficient to prove that $\mathsf{d}\text{-}\mathcal{P}^{\mathsf{NIKE}}$ does not exist in the sparse GGM/GBM. Due to space limit, we leave the proof in the full version of this paper.

5 Sparse GGM vs. EC-GGM

In this section, we analyze the relationship between the Elliptic Curve Generic Group Model (EC-GGM) and the sparse GGM within the framework of indifferentiability.

5.1 EC-GGM Statistically Implies Sparse GGM

We describe how to build a sparse generic group from an EC-GGM equipped with an independent random permutation. Denote Orac as the tuple $(\mathsf{ec}\text{-}\mathcal{G}_N, \mathsf{Perm})$, where $\mathsf{ec}\text{-}\mathcal{G}_N = (\mathsf{ec}\text{-}\mathcal{G}_N^{\mathsf{label}}, \mathsf{ec}\text{-}\mathcal{G}_N^{\mathsf{add}})$ is an elliptic curve generic group model over a point set E, and Perm is a random permutation over $\{0,1\}^m$. Define $\Delta m := m - (\lceil \log p \rceil + 1)$, the construction $\Pi^{\mathsf{Orac}} = (\mathsf{L}^{\mathsf{Orac}}, \mathsf{A}^{\mathsf{Orac}})$ is depicted in Fig. 4.

Theorem 5.1. *Let* $\mathcal{G}_{N,m}$ *be a sparse generic group model. The scheme* Π^{Orac} *is indifferentiable from* $\mathcal{G}_{N,m}$. *More precisely, there exists a simulator* $\mathcal{S}$ *such that for all* q*-query differentiator* $\mathcal{D}$, *we have*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi^{\mathsf{Orac}}, \mathcal{G}_{N,m}, \mathcal{S}, \mathcal{D}} \leq \frac{36q^2}{N} + \frac{12q^2 + q}{2^{\lambda}} + \frac{q}{2^{\omega(\log\lambda)}}.$$

The simulator makes at most λq *queries to* $\mathcal{G}_{N,m}$.

Due to space limit, we leave the proof in the full version of this paper.

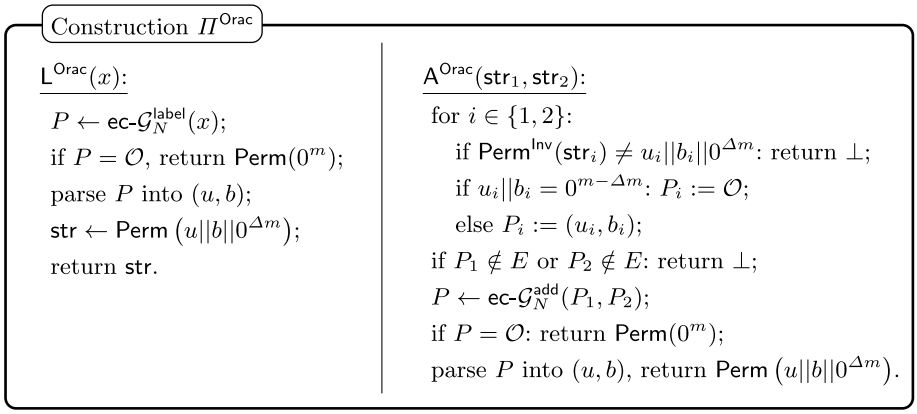

Fig. 4. The construction Π^{Orac} in $\mathsf{ec}\text{-}\mathcal{G}_N$ and Perm.

5.2 Sparse GGM Does Not Computationally Imply EC-GGM

Theorem 5.2. *Let $\mathcal{G}_{N,m}$ be a sparse generic group model, and let $\mathsf{ec}\text{-}\mathcal{G}_N$ be an elliptic curve generic group model over a point set E defined by the equation $y^2 = F(u)$ over $\mathbb{Z}_p$. Then, $\mathsf{ec}\text{-}\mathcal{G}_N$ is computationally indifferentiably separated from $\mathcal{G}_{N,m}$.*

Our strategy fundamentally builds on the analytical framework developed by Zhandry and Zhang [ZZ23]. Due to space limit, we leave the proof in the full version of this paper.

Acknowledgment. Cong Zhang was supported by the National Key Research and Development Program of China (Grant No. 2023YFB3106000). This work was supported by the Open Research Fund of The State Key Laboratory of Blockchain and Data Security, Zhejiang University This work was supported by Ant Group through CCF-Ant Research Fund (Grant No. CCF-AFSG RF20230308). Hong-Sheng Zhou was supported in part by NSF grant CNS-1801470 and a VCU Research Quest grant.

References

ABD+13. Andreeva, E., Bogdanov, A., Dodis, Y., Mennink, B., Steinberger, J.P.: On the indifferentiability of key-alternating ciphers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. Part I, volume 8042 of LNCS, pp. 531–550. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_29

Adl79. Adleman, L.: A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In: 20th Annual Symposium on Foundations of Computer Science (SFCS 1979), pp. 55–60. IEEE Computer Society (1979)

BB04. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_4

BBB+18. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334. IEEE Computer Society Press (May 2018)

BCI+10. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Berlin, Heidelberg (2010)

BF03. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)

BF18. Barbosa, M., Farshim, P.: Indifferentiable authenticated encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. Part I, volume 10991 of LNCS, pp. 187–220. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_7

BGOS07. Barreto, P.S.L.M., Galbraith, S.D., O'hEigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. DCC **42**(3), 239–271 (2007)

BKSY11. Brakerski, Z., Katz, J., Segev, G., Yerukhimovich, A.: Limits on the power of zero-knowledge proofs in cryptographic constructions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 559–578. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_34

CDMP05. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: how to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11535218_26

CDMS10. Coron, J.-S., Dodis, Y., Mandal, A., Seurin, Y.: A domain extender for the ideal cipher. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 273–289. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_17

CHK+16. Coron, J.-S., Holenstein, T., Künzler, R., Patarin, J., Seurin, Y., Tessaro, S.: How to build an ideal cipher: the indifferentiability of the Feistel construction. J. Cryptol. **29**(1), 61–114 (2016)

CMR+23. Chen, L., Moody, D., Randall, K., Regenscheid, A., Robinson, A.: Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters (2023). https://doi.org/10.6028/NIST.SP.800-186

DH76. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)

DHH+21. Döttling, N., Hartmann, D., Hofheinz, D., Kiltz, E., Schäge, S., Ursu, B.: On the impossibility of purely algebraic signatures. In: Nissim, K., Waters, B. (eds.) TCC 2021. Part III, volume 13044 of LNCS, pp. 317–349. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90456-2_1

DRS09. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging Merkle-Damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_22

DSSL16. Dodis, Y., Stam, M., Steinberger, J.P., Liu, T.: Indifferentiability of confusion-diffusion networks. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. Part II, volume 9666 of LNCS, pp. 679–704. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_24

ElG85. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)

FR94. Frey, G., Rück, H.-G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comput. **62**(206), 865–874 (1994)

FT10. Fouque, P.-A., Tibouchi, M.: Estimating the size of the image of deterministic hash functions to elliptic curves. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 81–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14712-8_5

Gro16. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. Part II, volume 9666 of LNCS, pp. 305–326. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11

GS22. Groth, J., Shoup, V.: On the security of ECDSA with additive key derivation and presignatures. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. EUROCRYPT 2022. LNCS, vol. 13275. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06944-4_13

GWL23. Guo, C., Wang, L., Lin, D.: Impossibility of indifferentiable iterated blockciphers from 3 or less primitive calls. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. Part IV, volume 14007 of LNCS, pp. 408–439. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30634-1_14

HMQS23. Hajiabadi, M., Mahmoody, M., Qi, W., Sarfaraz, S.: Lower bounds on assumptions behind registration-based encryption. In: Rothblum, G., Wee, H. (eds.) Theory of Cryptography. TCC 2023. LNCS, vol. 14370. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-48618-0_11

Ica09. Icart, T.: How to hash into elliptic curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_18

IR89. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC, pp. 44–61. ACM Press (May 1989)

JZW+24. Ji, K., Zhang, C., Wang, T., Zhang, B., Zhou, H.-S., Wang, X., Ren, K.: On the complexity of cryptographic groups and generic group models. In: Chung, K.-M., Sasaki, Yu. (eds.) ASIACRYPT 2024. Part VII, volume 15490 of LNCS, pp. 3–35. Springer, Singapore (2024). https://doi.org/10.1007/978-981-96-0941-3_1

Kob87. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)

LPS23. Lipmaa, H., Parisella, R., Siim, J.: Algebraic group model with oblivious sampling. In: Rothblum, G., Wee, H. (eds.) Theory of Cryptography. TCC 2023. LNCS, vol. 14372. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-48624-1_14

LZ24. George, L., Zhandry, M.: Limits on the power of prime-order groups: Separating Q-type from static assumptions. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024. Part V, volume 14924 of LNCS, pp. 46–74. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-68388-6_3

Mau05. Maurer, U.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 1–12. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11586821_1

Mil86. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO'85. LNCS, vol. 218, pp. 417–426. Springer, Berlin, Heidelberg (1986). https://doi.org/10.1007/3-540-39799-X_31

MR19. Masny, D., Rindal, P.: Endemic oblivious transfer. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019, pp. 309–326. ACM Press (November 2019)

MRH04. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_2

RSS11. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_27

RTV04. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_1

Sho97. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Berlin, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18

Zha22. Zhandry, M.: To label, or not to label (in generic groups). In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. Part III, volume 13509 of LNCS, pp. 66–96. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15982-4_3

ZZ20. Zhandry, M., Zhang, C.: Indifferentiability for public key cryptosystems. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. Part I, volume 12170 of LNCS, pp. 63–93. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_3

ZZ23. Zhang, C., Zhandry, M.: The relationship between idealized models under computationally bounded adversaries. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023. Part VI, volume 14443 of LNCS, pp. 390–419. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-8736-8_13