Preprocessing Security in Multiple Idealized Models with Applications to Schnorr Signatures and PSEC-KEM

Jeremiah Blocki
Purdue University
West Lafayette, USA
jblocki@purdue.edu

Seunghoon Lee
Purdue University
West Lafayette, USA
lee2856@purdue.edu

Abstract

In modern cryptography, relatively few instantiations of foundational cryptographic primitives are used across most cryptographic protocols. For example, elliptic curve groups are typically instantiated using P-256, P-384, Curve25519, or Curve448, while block ciphers are commonly instantiated with AES, and hash functions with SHA-2, SHA-3, or SHAKE. This limited diversity raises concerns that an adversary with nation-state-level resources could perform a preprocessing attack, generating a hint that might later be exploited to break protocols built on these primitives. It is often notoriously challenging to analyze and upper bound the advantage of a preprocessing attacker even if we assume that we have idealized instantiations of our cryptographic primitives (ideal permutations, ideal ciphers, random oracles, generic groups). Coretti et al. (CRYPTO/EUROCRYPT'18) demonstrated a powerful framework to simplify the analysis of preprocessing attacks, but they only proved that their framework applies when the cryptographic protocol uses a single idealized primitive. In practice, however, cryptographic constructions often utilize multiple different cryptographic primitives.

We verify that Coretti et al. (CRYPTO/EUROCRYPT'18)'s framework extends to settings with multiple idealized primitives and we apply this framework to analyze the multi-user security of (short) Schnorr Signatures and the CCA-security of PSEC-KEM against preprocessing attackers in the Random Oracle Model (ROM) plus the Generic Group Model (GGM). Prior work of Blocki and Lee (EUROCRYPT'22) used complicated compression arguments to analyze the security of *key-prefixed* short Schnorr signatures where the random oracle is salted with the user's public key. However, the security analysis did not extend to standardized implementations of Schnorr Signatures (e.g., BSI-TR-03111 or ISO/IEC 14888-3) which do not adopt key-prefixing, but take other measures to protect against preprocessing attacks by disallowing signatures that use a preimage of 0. Blocki and Lee (EUROCRYPT'22) left the (in)security of such "nonzero Schnorr Signature" constructions as an open question. We fully resolve this open question demonstrating that (short) nonzero Schnorr Signatures are also secure against preprocessing attacks. We also analyze PSEC-KEM in the ROM+GGM demonstrating that this Key Encapsulation Mechanism (KEM) is CPA-secure against preprocessing attacks.

Keywords

Short Schnorr Signatures, Key Encapsulation Mechanism, PSEC-KEM, Preprocessing Attacks, Bit-Fixing Model

1 Introduction

Many cryptographic protocols can be designed and analyzed from a handful of primitives. For example, the Schnorr signature scheme [38] uses an elliptic curve group and a cryptographic hash function while the simple oblivious transfer protocol [10] uses an elliptic curve group, a cryptographic hash function, and a block cipher. In practice, there are only a small number of viable instantiations of each of these cryptographic primitives, e.g., AES, Triple DES, SHA-2, SHA-3, SHAKE, P-256, P-384, Curve25519, Curve448, and so on. This raises the concern that an attacker with nation-state-level resources might attempt to execute a *preprocessing attack*. A preprocessing attacker invests massive computational resources to generate a hint in the hope that this hint will help an online attacker quickly break cryptographic protocols that use the same instantiations of our cryptographic primitives.

There is a line of prior work considering preprocessing attacks on various cryptographic primitives and underlying problems, e.g., on the discrete-log problem [5, 9, 11, 14, 30, 37], on the distributed discrete-log problem [28], on block ciphers [17, 27, 45] on the Computational/Decisional Diffie-Hellman problem [14, 37], on the salted Merkle-Damgård construction [2, 3, 12, 25, 26], on Sponge hashing [11, 24], on Schnorr signatures [6, 18], and much more. For example, Corrigan-Gibbs and Kogan [14] proved that any preprocessing attacker succeeds in solving the discrete log problem with probability at most $\widetilde{O}(Sq^2/p)$ where p is the size of the generic group, S is the size of the hint generated by the offline attacker, and q is the total number of online queries and they also gave a generic attack that matched this upper bound.

Many of these preprocessing lower bounds [2, 6, 14, 15, 17] utilized complex and technically challenging compression arguments which made it difficult to easily extend the ideas to new problems. Coretti et al. [12] introduced a comprehensive framework aimed at simplifying security analyses in the context of preprocessing in the Random Oracle Model (ROM) [4] by proving improved bounds for Unruh's pre-sampling technique [44]. In particular, Coretti et al. [12] showed how to translate the security in the Bit-Fixing ROM (BF-ROM) [44] into the security in the Auxiliary-Input ROM (AI-ROM). In the BF-ROM, the preprocessing attacker is allowed to fix P random input/output pairs for the random oracle and the remaining outputs are selected uniformly at random, whereas in the AI-ROM, the preprocessing attacker, after interacting with the random oracle (possibly making every possible query), generates a S-bit hint for the online attacker. While the BF-ROM is not compelling as a realistic model of preprocessing attacks in the real world, it is often significantly easier to upper bound the attacker's advantage in the BF-ROM. For example, consider the function inversion problem where we are given a random string $y \in \{0,1\}^\lambda$ and the goal is to find an input x such that $f(x) = y$. In the BF-ROM, it is trivial to upper bound the attacker's success probability as $(P + q)/2^\lambda$ where q denotes the number of random oracle queries made by the online attacker. The classical Auxiliary-Input ROM (AI-ROM) is more realistic model of real-world attacks, but proving lower bounds in this model often involves the use of notoriously challenging compressing arguments. Coretti et al. [12] showed how upper bounds on the attacker's advantage in the BF-ROM $\varepsilon_{\mathsf{BF},P,q}$ imply upper bounds in the AI-ROM $\varepsilon_{\mathsf{AI},S,q}$. In particular, if any bit-fixing attacker who can fix P points in the offline phase and make q queries in the online phase succeeds with probability at most $\varepsilon_{\mathsf{BF},P,q}$ then any AI-ROM attacker utilizing S-bit hints and making q online queries succeeds with probability at most $\varepsilon_{\mathsf{AI},S} \leq \varepsilon_{\mathsf{BF},P,q} + O(Sq/P)$. As a concrete example, this immediately implies a lower bound of $O\left(\sqrt{Sq/2^\lambda}\right)$ for the function inversion problem in the AI-ROM by setting $P = \sqrt{Sq2^\lambda}$. This matches state-of-the-art lower bounds [2] for the function inversion problem obtained using complicated compression arguments. Coretti et al. [11] extended the BF-ROM framework [12] to encompass other idealized primitives including the Generic Group Model (GGM) [33, 42], Ideal Cipher Model (ICM), and Random Permutation Model (RPM). They proved that (1) BF-GGM lower bounds yield AI-GGM lower bounds with a similar additive loss $O(Sq/P)$, (2) BF-ICM lower bounds yield AI-ICM lower bounds with a similar additive loss $O(Sq/P)$, and (3) BF-RPM lower bounds yield AI-RPM lower bounds with a similar additive loss $O(Sq/P)$.

Technically, the results in Coretti et al. [11, 12] are only stated for a *single* idealized primitive, while many cryptographic constructions utilize several primitives, e.g., Schnorr signatures and PSEC-KEM (Provably Secure Elliptic Curve-Key Encapsulation Mechanism; an El-Gamal-based key encapsulation mechanism developed by NTT [41, 43]) both utilize elliptic curve groups and cryptographic hash functions. Thus, if we want to analyze the security of these constructions against preprocessing attacks, we will be working with *multiple* idealized primitives. For example, Blocki and Lee [6] analyzed the multi-user security of (short) Schnorr signatures against preprocessing attackers in the ROM+GGM. In particular, they established the multi-user security of *key-prefixed* Schnorr signatures against preprocessing attackers — key-prefixed Schnorr Signatures salt the random oracle with the user's public key. However, their analysis involved a complicated compression argument, and for technical reasons, they needed to place a time-restriction on the preprocessing attacker so that the preprocessing attack can only examine the random oracle on a negligibly small fraction of inputs (e.g., the attacker can only examine $2^{3\lambda}$ of the total possible $2^{4\lambda}$ input/output pairs). The security proof also did not extend to the original Schnorr signature scheme without key-prefixing since this scheme is trivially broken by preprocessing attacks who can find a pre-image of 0. In particular, the preprocessing attacker can simply output a pair (r, m) such that $\mathsf{H}(g^r \| m) = 0$ so that $\sigma = (r, 0)$ becomes a universal signature for the message m which will validate under *any* public key! Standardized implementations of Schnorr Signatures (e.g., BSI-TR-03111 [21] or ISO/IEC 14888-3 [23]) do not adopt key-prefixing, but explicitly disallow those "$e = 0$ signatures". We refer to such implementations as *Nonzero Schnorr Signatures.* Thus, it is unclear whether or not Nonzero Schnorr Signatures are secure against preprocessing attacks. Establishing the (in)security of Nonzero Schnorr Signatures against preprocessing attacks was left as an open research direction [6]. This leads us to ask the following questions:

- *Does Coretti et al.'s BF-to-AI framework extend to settings with* multiple *idealized models?*
- *Do (short) key-prefixed Schnorr Signatures remain secure against preprocessing attackers if we do not limit the attacker's running time during the preprocessing phase?*
- *Are standardized implementations of Schnorr signatures (no key-prefixing, $e = 0$ signatures explicitly disallowed) secure against preprocessing attacks?*
- *Can we establish the preprocessing security of other cryptographic protocols, e.g., PSEC-KEM, which utilize multiple idealized models?*

1.1 Our Contributions

We first verify that the results of Coretti et al. [11, 12] extend to settings with *multiple* idealized models. In hindsight this result is not particularly surprising, but Coretti et al. [11, 12] never consider this setting so it is necessary to verify that the framework still applies. We show that the framework actually allows us to analyze constructions and protocols that utilize multiple random oracles, generic group oracles, ideal ciphers, and random permutations. In particular, we verify that Bit-Fixing upper bounds yield upper bounds on the attacker's success rate in the Auxiliary-Input model with an additive loss $O(Sq/P)$. Here, S denotes the size of the hint generated by the offline attacker in the Auxiliary-Input model, P denotes the total number of input/output pairs fixed by our Bit-Fixing attacker across all idealized primitives and q denotes the total number of oracle queries made by the online attacker across all idealized models.

Theorem 1.1 (informal). *If an application G is secure with probability ε' in the Bit-Fixing multiple idealized models (e.g., ROM+GGM+ICM) which fixes P coordinates of random oracles/generic group oracles/ideal ciphers in total, then it is also secure with probability ε in the corresponding Auxiliary-Input multiple idealized model with an S-bit hint, with $\varepsilon \leq \varepsilon' + O(Sq/P)$ where q is the combined online query complexity corresponding to G that corresponds to the random oracles/generic group oracles/ideal ciphers (see Definition B.1 and Theorem B.4).*

This resolves our first question in the affirmative. In the formal statement of the theorem (Theorem B.4), we establish slightly more refined bounds when we have separate upper bounds on the number of queries to each ideal oracle. Note that in the Auxiliary-Input model, the S-bit hint could encode information about relationships between the various ideal oracles, e.g., the hint could identify two generic group elements that hash to the same value under our random oracle. Thus, we cannot simply apply the results of Coretti et al. [11] separately to each idealized primitive in a black-box manner. However, the proof of Theorem B.4 closely follows prior analysis of Coretti et al. [11, 12] so in hindsight the final result is not particularly "surprising." In any case it is useful to formally verify that the framework extends to settings with multiple idealized models.

Application 1: Multi-User Security of Short Nonzero Schnorr Signatures Against Preprocessing Attacks. We leverage Theorem 1.1 to analyze the multi-user security of key-prefixed short Schnorr signatures and *regular (non-key-prefixed)* short *nonzero* Schnorr signatures in the ROM+GGM, answering our second and third question in the affirmative. Here, we use *nonzero Schnorr signatures* to refer to implementations where we explicitly disallow $e = 0$ signatures (see Definition 4.1 for the formal definition). We first upper bound the success rate of a multi-user signature forgery attacker in the Bit-Fixing model and then leverage Theorem 1.1 to obtain upper bounds in the Auxiliary-Input model where the preprocessing attacker outputs an S-bit hint after performing an *unbounded* number of queries to the generic group oracles as well as the random oracles.

In the Bit-Fixing model, we are able to upper bound the attacker's advantage as $O\left(qP/p + q/2^{\lambda}\right)$ where p is the size of our generic group and λ denotes the number of output bits for our random oracle. This bound holds for *both* key-prefixed short Schnorr signatures and for nonzero Schnorr signatures without key-prefixing. We note that this upper bound on the advantage of a bit-fixing attacker is tight as a bit-fixing attacker can solve the discrete log problem with advantage $\Omega(qP/p)$. Now setting $P = Sq$, we can upper bound the advantage of an auxiliary-input attacker as $O\left(Sq^2/p + q/2^{\lambda}\right)$.

Theorem 1.2 (informal). *Any preprocessing attacker that is unbounded and outputs an S-bit hint during the preprocessing phase and makes at most q_{on} queries during the online phase wins the multi-user signature forgery game (chosen message attack) against the short nonzero Schnorr signature scheme with probability at most $O\left(Sq_{\mathsf{on}}^2/p + q_{\mathsf{on}}/2^{\lambda}\right)$ in the generic group model of order $p > 2^{2\lambda}$ plus programmable random oracle model (see Theorem 4.5). Essentially the same result holds for key-prefixed short Schnorr signatures as well (see Theorem C.2).*

We slightly oversimplified the above statement of Theorem 1.2 by hiding lower order terms involving the number of users N[1]. If we want to achieve λ-bit security[2] against a preprocessing attacker with an S-bit hint we need to set $p \geq 2^{2\lambda}S$ so that $Sq^2/p \leq q^2 2^{-2\lambda} \leq q2^{-\lambda}$.

Previous work [6] gave an upper bound of $O\left(Sq_{\mathsf{on}}^2 \log p/p + q_{\mathsf{on}}/2^{\lambda}\right)$ for short key-prefixed Schnorr signatures. Compared to this bound, we stress that (1) our bound is *tighter*, and furthermore (2) our result is more general in that we do not impose any limitation on the running time of the offline attacker. We remark that one can actually achieve λ-bit security for slightly smaller group size, i.e., we require $p \approx 2^{2\lambda}S$ instead of requiring $p \approx 2^{2\lambda}S \log p$. This is due to the missing $\log p$ term reduces the required signature length to 3.5λ (see discussion below) instead of $3.5\lambda + O(\log \lambda)$ [6]. Furthermore, we stress that our results also resolve an open question [6] left about the preprocessing security of nonzero Schnorr signatures.

[1]Terms involving N are dominated will all be lower-order terms unless the number of users is implausibly large, e.g., $N \geq \min\{2^{\lambda}, Sq\}$ where q is the total number of oracle queries made by the attacker.

[2]For "λ-bit security" we require that for any running time parameter $t \leq 2^{\lambda}$ that any attacker running in time t succeeds with probability at most $O\left(t/2^{\lambda}\right)$.

Theorem 1.2 provides valuable insights into the security of short nonzero Schnorr signatures. It demonstrates that by suitably selecting parameters, these signatures can achieve λ bits of multi-user security, even in the presence of preprocessing attacks. In particular, when setting $p \approx 2^{2\lambda}S$ and utilizing λ-bit hash outputs, the short nonzero Schnorr signatures maintain λ bits of multi-user security against our preprocessing attacker. For instance, if we choose $S = 2^{\lambda/2}$, then setting $p \approx 2^{2.5\lambda}$ results in signatures of length $\lambda + \log p \approx 3.5\lambda$ bits, which is shorter than that of regular Schnorr signatures (which require 4λ bits).

Application 2: Security of PSEC-KEM Against Preprocessing Attacks. Another application of our result is analyzing the preprocessing security of an El-Gamal-based key encapsulation mechanism called PSEC-KEM [41, 43] in the ROM+GGM, answering our final question in the affirmative as well. PSEC-KEM has several standardized implementation [1, 22, 41] and achieves provable security in the ROM plus the hardness assumption of the discrete logarithm problem [7, 32, 41], yet the preprocessing security of PSEC-KEM has never been studied. We show that PSEC-KEM is CPA-secure against preprocessing attacks. In particular, we prove that PSEC-KEM is CPA-secure in the Bit-Fixing ROM+GGM then apply Theorem 1.1 to prove CPA security in the Auxiliary-Input ROM+GGM. In the Bit-Fixing model, we upper bound the attacker's advantage as $1/2 + O\left(qP/p + qP/2^{\lambda_1} + q/2^{\lambda}\right)$ where λ_1 specifies the length of certain hash inputs/outputs in the PSEC-KEM protocol and p is the size of our generic group. Setting $P = Sq$, we obtain the following bound for the advantage of an auxiliary-input attacker.

Theorem 1.3 (informal). *Any preprocessing attacker that is unbounded and outputs an S-bit hint during the preprocessing phase and makes at most q_{on} queries during the online phase wins the CPA indistinguishability game against PSEC-KEM with probability at most $\frac{1}{2} + O\left(Sq_{\mathsf{on}}^2/p + Sq_{\mathsf{on}}^2/2^{\lambda_1} + q_{\mathsf{on}}/2^{\lambda}\right)$ in the generic group model of order $p > 2^{2\lambda}$ plus programmable random oracle model where λ_1 specifies the length of certain hash inputs/outputs in the PSEC-KEM protocol (see Theorem 5.4).*

A key ingredient in proving the preprocessing security of PSEC-KEM is via reduction from a new game called *the quadratic bridge-finding game*, where the goal of the attacker is to output a certain quadratic relationship between the unknown values (see Section 5.1). We prove that the probability to win the quadratic bridge-finding game for the bit-fixing attacker is upper bounded by $O(Pq/p)$. From Theorem 1.3, one could observe that PSEC-KEM achieves λ bits of *preprocessing* security by setting $p \approx 2^{2\lambda}S$ and setting the hash output length $\lambda_1 \approx 2\lambda + \log S$.

1.2 Related Work

The Random Oracle Model. The Random Oracle Model (ROM) is a type of an idealized model formalized by Bellare and Rogaway [4], in which we model a cryptographic hash function H as a truly random function.

The Generic Group Model. The Generic Group Model (GGM) is an idealized model proposed by Shoup [42]. One key motivation of the GGM is that it can be used to analyze several computational hardness assumptions. In particular, Shoup [42] proved that the discrete-log problem in a group of prime order p requires $\Omega(\sqrt{p})$ to solve, and the same lower bound holds for Computational Diffie-Hellman (CDH) problem and Decisional Diffie-Hellman (DDH) problem as well.

We remark that there are different GGMs; Schnorr and Jakobsson's model [39] where the attacker is not directly given a labeling map, Maurer's model [31] that uses pointers instead of random encoding of a group, and Kiltz et al.'s model [29] in which the generic group oracle maintains a global counter. While those models are incomparable in general, Zhandry [46] argued that Shoup's model should be preferred as Maurer's model fails to capture many generic techniques such as Merkle-Damgård transform, Feistel network, authenticated encryption, etc. Blocki and Lee [6] also noted that Schnorr and Jakobsson's model and Kiltz et al.'s model do not provide a natural way to analyze preprocessing security since the attacker is never given bit encodings of group elements which could be processed outside of the generic group oracles. Because we are focused on preprocessing attacks, we will utilize Shoup's GGM in this paper.

Equivalence of ROM and ICM. The Ideal Cipher Model (ICM) [40] and the Random Permutation Model (RPM) are widely used idealized models which often offer a simple framework to analyze the security of practical cryptographic protocols and constructions. It is known that the ROM and the ICM are equivalent by Coron et al. [13] in a sense that one can build an ideal cipher from a random oracle and vice versa. In particular, Coron et al. proved that a 6-round Luby-Rackoff Feistel network is indifferentiable from a random permutation. Given that one can construct ideal ciphers from random oracles (or vice versa) one may wonder whether it is necessary to develop tools to analyze preprocessing security in multiple different idealized models. We note that the concrete security guarantee is not ideal as the distinguishing advantage is upper bounded as $O(q^{16}/2^{\lambda})$. Dai and Steinburger [16] improved this bound to $O(q^{8}/2^{\lambda})$, but one would still need to select a much larger security parameter, e.g., $\lambda = 8\lambda'$ achieves λ'-bit security. Furthermore, the indifferentiability results [13, 16] do not consider preprocessing attackers and it is unknown whether these indifferentiability results can be extended to the Bit-Fixing model or to the Auxiliary-Input model. Thus, it is still useful to develop tools to establish preprocessing security with multiple idealized models.

Short Schnorr Signatures. Schnorr's original paper [38] proposed the idea of shortening Schnorr Signatures by truncating the hash output. The security of the Short Schnorr Signature scheme has been subsequently studied under a variety of assumptions/settings, e.g., see [6, 29, 34, 39], but only [6] considers preprocessing security. We compare our results with [6] extensively throughout the paper.

Preprocessing Attacks and Lower Bounds. There is a long line of work analyzing the security of various cryptogrphic primitivies against preprocessing attacks, e.g., [2, 5, 9, 11, 14, 15, 17, 24, 27, 30, 44, 45]. The presampling framework of Coretti et al. [11, 12] is most directly related to our paper and is discussed in Section 3.

2 Preliminaries

Let $\mathbb{N}$ be the set of positive integers, and we define $[N] \coloneqq \{1, \ldots, N\}$ for $N \in \mathbb{N}$. Throughout the paper, we denote the security parameter by λ. We define $\mathbb{Z}_p$ to be a cyclic group of integers modulo p. We say that $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{Z}_p^N$ is an N-dimensional vector over $\mathbb{Z}_p^N$, and for each $i \in [N]$, we define $\hat{u}_i$ to be the i^{th} N-dimensional unit vector, i.e., the i^{th} element of $\hat{u}_i$ is 1, and all other elements are 0 elsewhere. For simplicity, we let $\log(\cdot)$ be a log with base 2, i.e., $\log x \coloneqq \log_2 x$. The notation $\hookleftarrow\$$ denotes a uniformly random sampling, e.g., we say $x \hookleftarrow\$ \mathbb{Z}_p$ when x is sampled uniformly at random from $\mathbb{Z}_p$.

Operations in Idealized Models. In the ROM, the only way for the adversary to compute $\mathsf{H}(x)$ is querying x to H as an oracle. If x has not been queried before, then the value of $\mathsf{H}(x)$ is uniformly at random.

As the name suggests, the GGM only allows the adversary to have access to a random representation of a group which is done by a randomly chosen encoding (labeling) so that the adversary can only make the group operations in a black-box manner. More precisely, given a cyclic group $G = \langle g \rangle$ of prime order p, a random injective encoding (labeling) map $\tau : \mathbb{Z}_p \to \mathbb{G}$, where $\mathbb{G}$ is the set of bit strings of length $\ell \geq \lceil \log p \rceil$, and we encode the discrete log of a group element instead of the group element itself for simplicity. In the GGM, the adversary is only given access to oracles $\mathtt{Mult}(\cdot,\cdot)$ and $\mathtt{Inv}(\cdot)$ which computes the group operation *indirectly* in G, as well as the encoding of the generator of G, i.e., $\mathfrak{g} = \tau(1)$. In particular, for $(\mathfrak{a}, \mathfrak{b}) \in \mathbb{G} \times \mathbb{G}$, the oracles work as $\mathtt{Mult}(\mathfrak{a}, \mathfrak{b}) = \tau(\tau^{-1}(\mathfrak{a}) + \tau^{-1}(\mathfrak{b}))$ and $\mathtt{Inv}(\mathfrak{a}) = \tau(-\tau^{-1}(\mathfrak{a}))$ if $\mathfrak{a}, \mathfrak{b} \in \tau(G)$.[3]

In the RPM, all parties have oracle access to a uniform permutation P and its inverse P^{-1}, i.e., $P^{-1}(P(x)) = x$. The RPM has been used as a framework to analyze key-alternating ciphers (e.g., AES [36]), Even-Mansour cipher [20], and sponge-based constructions such as SHA-3 [19, 35]. In the ICM which goes back to Shannon [40], we assume that the block cipher *is* a random permutation for *every* key K. In particular, for any fixed key K, the function $E_K(x) \coloneqq E(K, x)$ is a truly random permutations with inverse $E_K^{-1}(y) \coloneqq E^{-1}(K, y)$, i.e., $E_K^{-1}(E_K(x)) = x$. In the ICM, all parties (honest parties and adversaries included) have oracle access to the block cipher $E(\cdot,\cdot)$ and its inverse $E^{-1}(\cdot,\cdot)$. If Alice and Bob share a secret key K then they can both evaluate the permutation $E_K(\cdot)$ and its inverse $E_K^{-1}(\cdot)$, but a party who does not have K will not be able to perform this computation using the block cipher oracles $E(\cdot,\cdot)$ or $E^{-1}(\cdot,\cdot)$.

3 Background: Bit-Fixing/Auxiliary-Input ROM+GGM and Extension to Multiple Idealized Models

Coretti et al. [11, 12] introduced a breakthrough technique that generically translates the lower bound in the Bit-Fixing idealized model (e.g., ROM, GGM, ICM, or RPM) to the lower bound in the corresponding Auxiliary-Input model with optimal additive security loss. Bit-Fixing model was first introduced by Unruh [44] and formalized by Coretti et al. [12]. In the Bit-Fixing model with a parameter $P \in \mathbb{N}$, P arbitrary input/output pairs can be fixed in advance and the rest of the coordinates are chosen at random in

[3]We can also define an oracle $\mathsf{Pow}(\mathfrak{a}, n) \coloneqq \tau(n\tau^{-1}(\mathfrak{a}))$ for convenience, which can be computed by making $O(\log n)$ oracle calls to $\mathtt{Mult}$ using the standard modular exponentiation algorithm.

the preprocessing phase, which is independent of the fixed pairs. Furthermore, the S-bit hint of the preprocessing attacker in the Bit-Fixing model can only depend on those P fixed points. On the other hand, in the Auxiliary-Input model, the preprocessing-phase attacker is unbounded and outputs an S-bit hint for the online attacker after viewing the entire truth table of the corresponding oracle, which characterizes preprocessing attacks in a more feasible way.

Even though the Bit-Fixing model is not a compelling model for capturing preprocessing attacks, it happens to be much easier to prove lower bounds in the Bit-Fixing model instead of the Auxiliary-Input model which often times uses a complicated compression argument, e.g., see [14]. Furthermore, since there exists a generic translation of the lower bound in the Bit-Fixing model to the Auxiliary-Input model [11, 12], we can conduct a simpler analysis in the Bit-Fixing model and make a transition to the Auxiliary-Input model.

While prior work focused on a *single* idealized model, there are several cryptographic primitives that rely on *multiple* idealized models; for instance, the security analysis of short Schnorr signatures and PSEC-KEM relies on both the Random Oracle Model plus the Generic Group Model [6], and the security of a simple oblivious transfer protocol [10] even relies on three different idealized models — Random Oracle Model, Generic Group Model, and Ideal Cipher Model. Hence, it is crucial to have a generic translation from a Bit-Fixing model with *multiple* idealized models to an Auxiliary-Input model with the corresponding idealized models. As a warmup, we consider the ROM+GGM which can be used to analyze the multi-user security of Schnorr signatures against preprocessing attacks and the security of an El-Gamal-based key encapsulation mechanism.

Informal Definition of the Bit-Fixing (BF) ROM+GGM. We use the notation $\mathsf{BF\text{-}RO{+}GG}(P_1, P_2, m, \lambda, p, \ell)$ to denote the Bit-Fixing model with a random oracle $\mathsf{H} : \{0,1\}^m \to \{0,1\}^\lambda$ and a generic group of size p with group elements encoded as ℓ-bit strings using a random injective map $\tau : \mathbb{Z}_p \to \{0,1\}^\ell$ where $\ell \geq \lceil \log_2 p \rceil$. In this model, the preprocessing attacker $\mathcal{A}_{\mathsf{pre}}$ may fix P_1 (resp. P_2) input/output pairs for the random oracle H (resp. injective map τ) and then output an S-bit hint for the online attacker $\mathcal{A}_{\mathsf{on}}$ to use. However, the remaining input/output pairs are picked randomly (resp. randomly subject to the constraint that τ is still an injective map) and entirely uncorrelated with the S-bit hint output by the preprocessing attacker. We say that an application is (S, T_1, T_2, ϵ)-secure in the $\mathsf{BF\text{-}RO{+}GG}(P_1, P_2, m, \lambda, p, \ell)$-model if any attacker $\mathcal{A} = (A_{\mathsf{pre}}, A_{\mathsf{on}})$ wins the security game for our application with probability at most ϵ. Here, we constrain the online attacker to make at most T_1 (resp. T_2) random oracle queries (resp. generic group queries). As before, we constrain the preprocessing attacker $\mathcal{A}_{\mathsf{pre}}$ to fix at most P_1 (resp. P_2) points in the random oracle (resp. injective map τ) and to output a hint of length at most S bits. While the Bit-Fixing model is not a particularly compelling model of preprocessing attacks, it is a useful tool to simplify security analysis in more realistic settings such as the Auxiliary-Input model (AI). See Section A.2 for a more formal definition of the Bit-Fixing ROM+GGM.

Informal Definition of the Auxiliary-Input (AI) ROM+GGM. We use the notation $\mathsf{AI\text{-}RO{+}GG}(m, \lambda, p, \ell)$ to denote the Auxiliary-Input model with a random oracle $\mathsf{H} : \{0,1\}^m \to \{0,1\}^\lambda$ and a generic group of size p with group elements encoded as ℓ-bit strings using a random injective map $\tau : \mathbb{Z}_p \to \{0,1\}^\ell$ where $\ell \geq \lceil \log_2 p \rceil$. In this model, the preprocessing attacker $\mathcal{A}_{\mathsf{pre}}$ may examine the entire input/output tables of the random oracle H and the injective map τ and then output an S-bit hint for the online attacker $\mathcal{A}_{\mathsf{on}}$. We say that an application is (S, T_1, T_2, ϵ)-secure in the $\mathsf{AI\text{-}RO{+}GG}(m, \lambda, p, \ell)$-model if any preprocessing attacker $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$ wins the security game for our application with probability at most ϵ. Here, we constrain the online attacker to make at most T_1 (resp. T_2) random oracle queries (resp. generic group queries) and, as before, we constrain the offline attacker $\mathcal{A}_{\mathsf{pre}}$ to output an S-bit hint. See Section A.2 for a more formal definition of the Auxiliary-Input ROM+GGM.

Additive Error for Arbitrary Applications in the ROM+GGM. The main result considering BF-ROM+GGM to AI-ROM+GGM transition is stated in Theorem 3.1 below. The security bound references the combined query complexity of our application G. Intuitively, $(T_1)_G^{\mathsf{comb}}$ (resp. $(T_2)_G^{\mathsf{comb}}$) denotes the total number of random oracle (resp. generic group) queries made by our online attacker $\mathcal{A}_{\mathsf{on}}$ as well as by the challenger in the security game for G. In all of our applications, combined query complexity is not substantially larger than the query complexity of the online attacker, i.e., $(T_i)_G^{\mathsf{comb}} = O(T_i)$ for $i = 1, 2$.

Theorem 3.1. *For every $\gamma > 0$, if an application G is $(S, T_1, T_2, \varepsilon')$-secure in the $\mathsf{BF\text{-}RO{+}GG}(P_1, P_2, m, \lambda, p, \ell)$-model for any $P_1, P_2 \in \mathbb{N}$ such that $P_1\lambda + P_2 \log(p/e) \leq \eta$, then it is $(S, T_1, T_2, \varepsilon)$-secure in the $\mathsf{AI\text{-}RO{+}GG}(m, \lambda, p, \ell)$-model, for*

$$\varepsilon \leq \varepsilon' + \frac{2(S + \log \gamma^{-1}) \left((T_1)_G^{\mathsf{comb}} \lambda + 3(T_2)_G^{\mathsf{comb}} \log p \right)}{\eta} + 2\gamma,$$

where e is the Euler constant, and $(T_1)_G^{\mathsf{comb}}$ and $(T_2)_G^{\mathsf{comb}}$ are the combined query complexity corresponding to G that corresponds to the random oracle and the generic group oracle, respectively.

Proof Intuition. Our proof of Theorem 3.1 follows a similar approach to Coretti et al. [11, 12]. We first introduce the notion of a $(P_1, P_2, 1-\delta)$-dense source as a slight generalization of a $(P, 1-\delta)$-dense source [11, 12]. Intuitively, a $(P_1, P_2, 1-\delta)$-dense source is a pair of random variables (X, Y) where X (resp. Y) corresponds to a function H (resp. injective map τ) that is fixed on at most P_1 (resp. P_2) coordinates and for any subsets (I_1, I_2) of non-fixed inputs, the minimum entropy of the pair (X_{I_1}, Y_{I_2}) is $(1-\delta)$-close to the minimum entropy that we would have had if, on all of the remaining non-fixed points, H was a truly random function and τ was truly random injective function. Formally, we require

$$H_\infty(X_{I_1}, Y_{I_2}) \geq (1-\delta) \left[|I_1|\lambda + \log(p - P_2)^{\underline{|I_2|}} \right],$$

where $a^{\underline{b}} \coloneqq a!/(a-b)!$. See Definition A.1 for a formal definition of dense sources. Following the strategy of Coretti et al. [11, 12], we then show (see Claim 1) that for every leaky source (i.e., the random oracle H and injective map τ after the preprocessing attacker provides an S-bit hint) is γ-close to a convex combination of sources such that each source in the convex combination is $(P_1', P_2', 1-\delta)$-dense for some P_1' and P_2' which satisfies

$$P_1'\lambda + P_2' \log(p/e) \leq \frac{S_z + \log \gamma^{-1}}{\delta}.$$

In Claim 2, we then upper bound the probability that a distinguisher can differentiate between a $(P_1', P_2', 1-\delta)$-dense source and the corresponding (P_1', P_2')-Bit-Fixing source. Taken in combination, this allows us to prove Theorem 3.1. See more details in Appendix A. □

Multiplicative Error for Unpredictability Applications in the ROM+GGM. Coretti et al. [12, Theorem 6] further proved that for any *unpredictability* application, the security bound from the BF-ROM translates into the AI-ROM at the cost of a multiplicative factor of 2. Here, we verify that a similar translation can be applied from the BF-ROM+GGM to the AI-ROM+GGM, which is formalized in the following Theorem. The proof of Theorem 3.2 can be found in Appendix A.

Theorem 3.2. *For every $\gamma > 0$, if an* unpredictability *application G is $(S, T_1, T_2, \varepsilon')$-secure in the* BF-RO+GG$(P_1, P_2, m, \lambda, p, \ell)$*-model for any $P_1, P_2 \in \mathbb{N}$ satisfying*

$$(S + \log \gamma^{-1}) \left((T_1)_G^{\text{comb}} \lambda + 3(T_2)_G^{\text{comb}} \log p \right) \leq P_1\lambda + P_2 \log(p/e),$$

then it is $(S, T_1, T_2, \varepsilon)$-secure in the AI-RO+GG$(m, \lambda, p, \ell)$*-model, for*

$$\varepsilon \leq 2\varepsilon' + 2\gamma,$$

where e is the Euler's number.

Generalization to Additional Idealized Models. The applications we consider in this paper only utilize the ROM+GGM, but other cryptographic constructions and protocols may utilize other primitives such as block ciphers or random permutations. For example, instantiations of protocols like TLS may utilize multiple hash functions (modeled as random oracles), elliptic curve groups (modeled as generic groups), and block ciphers (modeled as ideal ciphers). Thus, we generalize the results Theorem 3.1 and Theorem 3.2 to the setting where we have multiple different random oracles, generic groups, ideal ciphers, and ideal random permutations. Intuitively, we generalize the BF-ROM+GGM BF-RO+GG$(P_1, P_2, m, \lambda, p, \ell)$ to obtain a Bit-Fixing ROM+GGM+ICM BF-RO+GG+IC$(\mathbf{P}, \mathbf{Q}, \mathbf{R}, \{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$ where we have n_1 random oracles $\mathsf{H}_1, \ldots, \mathsf{H}_{n_1}$ with varying domains/ranges ($\mathsf{H}_i : [M_i] \to [N_i]$), n_2 Generic Groups of varying sizes with associated mappings $\tau_1, \ldots \tau_{n_2}$ ($\tau_i : \mathbb{Z}_{p_i} \to \{0,1\}^{\ell_i}$ for $\ell_i = \lceil \log_2 p_i \rceil$) and n_3 ideal ciphers $F_1, \ldots, F_{n_3}$ with varying keyspace $[K_i]$ and ciphertext space $[C_i]$[4]. The preprocessing attacker $\mathcal{A}_{\text{pre}}$ may fix at most $\mathbf{P}[i]$ (resp. $\mathbf{Q}[i]/\mathbf{R}[i]$) input/output pairs for the random oracle H_i (resp. injective mapping τ_i/ideal cipher F_i). The online attacker $\mathcal{A}_{\text{on}}$ is restricted to make at most $\mathbf{S}[i]$ (resp. $\mathbf{L}[i]$ and $\mathbf{T}[i]$) queries to the random oracle H_i (resp. to ideal cipher F_i and to the i^{th} generic group).

Our results indicate that if the total number of fixed points is bounded by W, i.e., $\|\mathbf{P}\|_1 + \|\mathbf{Q}\|_1 + \|\mathbf{R}\|_1 = \sum_i \mathbf{P}[i] + \sum_i \mathbf{Q}[i] + \sum_i \sum_j \mathbf{R}[i][j] \leq W$ then we can translate the security bound in the BF-ROM+GGM+ICM into the AI-ROM+GGM+ICM with an additive error $\widetilde{O}((S + \log \gamma^{-1})Q/W) + 2\gamma$ for any $\gamma > 0$, where Q denotes the total combined query complexity corresponding to an application G and $\widetilde{O}$ is hiding polylogarithmic factors. Furthermore, for unpredictability applications, we can obtain a similar result if $(S + \log \gamma^{-1})Q \leq C(\|\mathbf{P}\|_1 + \|\mathbf{Q}\|_1 + \|\mathbf{R}\|_1)$ for a certain constant $C > 0$. See Theorem 3.3 and Theorem 3.4 for simplified statements, and see Theorem B.4 and Theorem B.5 for the formal statements with tighter bounds. The proof of Theorem B.4 (resp. Theorem B.5) is similar to Theorem 3.1 (resp. Theorem 3.2) and can be found in Appendix B.

Theorem 3.3 (Informal Version of Theorem B.4). *For every $\gamma > 0$, if an application G is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon')$-secure in the* BF-RO+GG+IC$(\mathbf{P}, \mathbf{Q}, \mathbf{R}, \{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model for any $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ as introduced in Definition B.1 such that the total number of fixed points is bounded by W, i.e., $\|\mathbf{P}\|_1 + \|\mathbf{Q}\|_1 + \|\mathbf{R}\|_1 \leq W$. Then G is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon)$-secure in the* AI-RO+GG+IC$(\{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model, for $\varepsilon \leq \varepsilon' + \widetilde{O}\left(\frac{(S + \log \gamma^{-1})Q}{W} \right) + 2\gamma$, where Q denotes the total combined query complexity corresponding to G.*

Theorem 3.4 (Informal Version of Theorem B.5). *For every $\gamma > 0$, if an* unpredictability *application G is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon')$-secure in the* BF-RO+GG+IC$(\mathbf{P}, \mathbf{Q}, \mathbf{R}, \{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model for any $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ as introduced in Definition B.1 satisfying*

$$(S + \log \gamma^{-1})Q \leq \frac{\log(\min_i \{N_i, p_i/e, C_i/e\})}{\log(\max_i \{N_i, p_i, C_i\})} (\|\mathbf{P}\|_1 + \|\mathbf{Q}\|_1 + \|\mathbf{R}\|_1),$$

where e is the Euler's number and Q denotes the total combined query complexity corresponding to G. Then it is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon)$-secure in the AI-RO+GG+IC$(\{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model, for*

$$\varepsilon \leq 2\varepsilon' + 2\gamma.$$

4 Multi-User Security of Short Schnorr Signatures without Key-Prefixing against Preprocessing Attacks

We present the first application of our result (Theorem 3.1 and Theorem 3.2) from Section 3 to analyze the preprocessing security of short Schnorr signatures utilizing multiple idealized models. As Blocki and Lee [6] pointed out, Schnorr signatures (without key-prefixing) are vulnerable against a preprocessing attack by finding a pre-image of 0 for a hash function H, i.e., if a message m and an integer r are found during the preprocessing phase which satisfies $\mathsf{H}(\tau(r)\|m) = 0$, then the tuple (m, r) can simply be included in the S-bit hint. Then the signature $\sigma' = (s, e) = (r, 0)$ becomes valid for *any* public key pk which trivially breaks the signature scheme. To address this issue, Blocki and Lee considered the *key-prefixed* short Schnorr signatures by including the public key when computing e, i.e., $e = \mathsf{H}(\mathsf{pk}\|\tau(r)\|m)$. Since the preprocessing attacker does not have knowledge of the public key pk during the preprocessing phase, it becomes hard for the attacker to find a tuple (pk, r, m) such that $\mathsf{H}(\mathsf{pk}\|\tau(r)\|m) = 0$. Then they showed the multi-user security of *key-prefixed* short Schnorr signatures against preprocessing attacks.

On the other hand, instead of key-prefixing, several standardized implementations of Schnorr signatures, e.g., BSI-TR-03111 [21] and

[4]We do note separately model ideal permutations in our analysis as one can trivially obtain an ideal permutation from an ideal cipher e.g., fix $K = 0$ to obtain the ideal permutation $P(x) = E_0(x)$ and its inverse $P^{-1}(y) = E_0^{-1}(y)$.

ISO/IEC 14888-3 [23], resolved this issue by explicitly disallowing $e = 0$ signatures (which we call *nonzero Schnorr signatures*, see Definition 4.1) as that is the point where the vulnerability of Schnorr signatures to preprocessing attacks comes from. In this section, we analyze the multi-user security of nonzero Schnorr signatures.

```
Kg(1^λ):
1 : $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$
2 : $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
3 : return $(\mathsf{pk}, \mathsf{sk})$

Sign(sk, m):
1 : $r \leftarrow\!\!\$\ \mathbb{Z}_p$
2 : $I \leftarrow g^r$
3 : $e \leftarrow \mathsf{H}(I\|m)$
4 : while $e = 0$ do
5 :     Resample $r \leftarrow\!\!\$\ \mathbb{Z}_p$
6 :     $I \leftarrow g^r$
7 :     $e \leftarrow \mathsf{H}(I\|m)$
8 : $s \leftarrow r + \mathsf{sk} \cdot e \mod p$
9 : return $\sigma = (s, e)$

Vfy(pk, m, σ):
1 : Parse $\sigma = (s, e)$
2 : if $e = 0$ then
3 :     return 0
4 : $R \leftarrow g^s \cdot \mathsf{pk}^{-e}$
5 : if $\mathsf{H}(R\|m) = e$ then
6 :     return 1
7 : else return 0
```

Figure 1: The *nonzero* Schnorr signature scheme.

Definition 4.1. Given a cyclic group $G = \langle g \rangle$ and a prime p, the *nonzero Schnorr signature scheme* is a signature scheme that consists of a tuple of probabilistic polynomial-time algorithms $\Pi_{\setminus\{0\}} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$, where each algorithm works as follows:

- $\mathsf{Kg}(1^\lambda)$*:* a key-generation algorithm that takes 1^λ as input where λ is a security parameter and outputs a random secret key $\mathsf{sk} \in \mathbb{Z}_p$ and a public key $\mathsf{pk} = g^{\mathsf{sk}}$.
- $\mathsf{Sign}(\mathsf{sk}, m)$*:* a signing algorithm that takes the secret key sk and a message $m \in \{0,1\}^*$ as input and generates a signature $\sigma = (s, e)$ on the message m. When generating a signature, if $e = 0$ then the signing procedure regenerates the signature until we have $e \neq 0$. The signing procedure is described formally in Figure 1.
- $\mathsf{Vfy}(\mathsf{pk}, m, \sigma)$*:* a verification algorithm that outputs 1 if the generated signature is valid, and 0 otherwise. Here, we remark that in addition to the original verification algorithm that was proposed in Schnorr's original work [38], the algorithm also returns 0 if $\sigma = (s, e)$ is of the form $e = 0$. The verification procedure is described formally in Figure 1.

One could alternatively remove the loop in the signing procedure from Figure 1 and simply terminate and output $\perp$ with negligible probability. On the other hand, a more efficient way to generate a nonzero Schnorr signature is by outputting $e \leftarrow \mathsf{H}(I\|m) + 1$ on line 3 of $\mathsf{Sign}(\mathsf{sk}, m)$ in Figure 1. By doing this, the range of e lies in $\{1, \ldots, 2^{\lambda_1}\}$ and we have zero chance of $e = 0$. Then we do not even need to check if $e = 0$ in the verification procedure. And since $p > 2^{2\lambda} \gg 2^{\lambda_1}$, we do not need to worry about the overflow when adding 1 to $\mathsf{H}(I\|m)$.

1-out-of-N Generic Signature Forgery Game. To define the multi-user security of a signature scheme, recall the security game that was introduced by Blocki and Lee [6] for a signature scheme $\Pi = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ when we fix the injective mapping $\tau : \mathbb{Z}_p \to \mathbb{G}$, a random oracle H, and an adversary $\mathcal{A}$:

The 1-out-of-N Generic Signature Forgery Game $\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A},\Pi}(k)$:

1. $\mathsf{Kg}(1^k)$ is run N times to obtain the public and the secret keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each $i \in [N]$. Here, for each $i \in [N]$, sk_i is chosen randomly from the group $\mathbb{Z}_p$, where p is a $2k$-bit prime, and $\mathsf{pk}_i = \tau(\mathsf{sk}_i)$.
2. Adversary $\mathcal{A}$ is given $(\mathfrak{g} = \tau(1), \mathsf{pk}_1, \cdots, \mathsf{pk}_N, p)$, and access to the generic group oracles $\mathtt{GO} = (\mathtt{Mult}(\cdot,\cdot), \mathtt{Inv}(\cdot))$, the random oracle $\mathsf{H}(\cdot)$, and the signing oracles $\mathsf{Sign}(\mathsf{sk}_1, \cdot), \ldots, \mathsf{Sign}(\mathsf{sk}_N, \cdot)$. The experiment ends when the adversary outputs $(m, \sigma = (s, e))$.
3. $\mathcal{A}$ succeeds to forge a signature if and only if there exists some $j \in [N]$ such that $\mathsf{Vfy}(\mathsf{pk}_j, m, \sigma) = 1$ and the query m was never submitted to the oracle $\mathsf{Sign}(\mathsf{sk}_j, \cdot)$. The output of the experiment is $\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A},\Pi}(k) = 1$ when $\mathcal{A}$ succeeds; otherwise $\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A},\Pi}(k) = 0$.

Definition 4.2. Consider the generic group model with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$. A signature scheme $\Pi = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ is $(N, q_{\mathsf{H}}, q_{\mathsf{G}}, q_{\mathsf{S}}, \varepsilon)$*-MU-UF-CMA secure (multi-user unforgeable against chosen message attack)* if for every adversary $\mathcal{A}$ making at most q_{H} (resp. q_{G}, q_{S}) queries to the random oracle (resp. generic group, signing oracles), the following bound holds:

$$\Pr\left[\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A},\Pi}(k) = 1\right] \leq \varepsilon,$$

where the randomness is taken over the selection of τ, the random coins of $\mathcal{A}$, the random coins of Kg, and the selection of random oracle H.

When it comes to the *preprocessing security*, we say that a signature scheme Π achieves the multi-user security against preprocessing attacks if for every adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$, Π is $(N, q_{\mathsf{H}}, q_{\mathsf{G}}, q_{\mathsf{S}}, \varepsilon)$-MU-UF-CMA secure against the *online* attacker $\mathcal{A}_{\mathsf{on}}$. In the Bit-Fixing model, $\mathcal{A}_{\mathsf{on}}$ will receive the information of P pre-fixed points from $\mathcal{A}_{\mathsf{pre}}$, and in the Auxiliary-Input model, $\mathcal{A}_{\mathsf{on}}$ will receive an S-bit hint from $\mathcal{A}_{\mathsf{pre}}$. Hence, to prove the preprocessing security, it is sufficient to upper bound the probability of the event $\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A}_{\mathsf{on}},\Pi}(k) = 1$ for the online attacker $\mathcal{A}_{\mathsf{on}}$.

Multi-User Bridge-Finding Game in the BF-GGM. Blocki and Lee [6] established the multi-user security of short Schnorr signatures against preprocessing attackers by providing a reduction from a game called the 1*-out-of-N generic* BRIDGE^N*-finding game.* Given N inputs $\tau(x_1), \ldots, \tau(x_N)$, the goal of the game is to find a nontrivial linear dependence among those inputs, i.e., find $a_1, \ldots, a_N, b$ such that $\sum_{i=1}^N a_i x_i = b$, and $a_i \neq 0$ for at least one $i \in [N]$. The intuition here is that the probability that the attacker wins the 1-out-of-N discrete log game with (approximately) the same probability to win the 1-out-of-N generic BRIDGE^N-finding game [6, Corollary 1]. Since the odds to win the 1-out-of-N discrete log game is crucial to analyze the multi-user security of short Schnorr signatures, it is significant to analyze the probability to win the multi-user bridge-finding game.

In particular, analyzing the probability of winning the 1-out-of-N generic BRIDGE^N-finding game in the bit-fixing generic group model is essential when we do the reduction in analyzing the multi-user security of short Schnorr signatures in the Bit-Fixing model. Hence, we consider the 1-out-of-N generic BRIDGE^N-finding game in the BF-GGM which is described formally below.

The 1-out-of-N Generic BRIDGE^N-Finding Game $\mathrm{BridgeChal}_{\mathcal{A}}^{\tau,N}(\lambda,\mathbf{x})$ with a Bit-Fixing Preprocessing Attacker $\mathcal{A}=\left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)},\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}\right)$:

<u>Preprocessing Phase:</u>

1. $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ takes as input $\mathfrak{g}=\tau(1)$.
2. $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ fixes P input/output pairs of the map $\tau:\mathbb{Z}_p\to\mathbb{G}$, i.e., $(t_1,\tau(t_1)),\ldots,(t_P,\tau(t_P))$. (Note: the rest of the map τ is chosen uniformly at random as long as it remains to be injective.)

<u>Online Phase:</u>

3. The challenger initializes the list $\mathcal{L}=\{(\tau(1),\mathbf{0},1),(\tau(x_1),\hat{u}_1,0),\ldots,(\tau(x_N),\hat{u}_N,0),(\tau(t_1),\mathbf{0},t_1),\ldots,(\tau(t_P),\mathbf{0},t_P)\}$, and $\mathbf{x}=(x_1,\cdots,x_N)$.
4. The adversary $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ is given $\mathfrak{g}=\tau(1)$ and $\tau(x_i)$ for each $i\in[N]$.
5. $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ is allowed to query the usual generic group oracles (`Mult`, `Inv`).
   (a) If $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ ever submits any fresh element $\mathfrak{y}$ which does not appear in $\mathcal{L}$ as input to a generic group oracle, then the challenger immediately queries $b_y=\mathsf{DLog}(\mathfrak{y})$, and adds the tuple $(\mathfrak{y},\mathbf{0},b_y)$ to the list $\mathcal{L}$.
   (b) Whenever $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ submits a query $\mathfrak{y}_1,\mathfrak{y}_2$ to $\mathtt{Mult}(\cdot,\cdot)$, we are ensured that there exist tuples $(\mathfrak{y}_1,\mathbf{a}_1,b_1),(\mathfrak{y}_2,\mathbf{a}_2,b_2)\in\mathcal{L}$. The challenger adds the tuple $(\mathtt{Mult}(\mathfrak{y}_1,\mathfrak{y}_2),\mathbf{a}_1+\mathbf{a}_2,b_1+b_2)$ to the list $\mathcal{L}$.
   (c) Whenever $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ submits a query $\mathfrak{y}$ to $\mathtt{Inv}(\cdot)$, we are ensured that some tuple $(\mathfrak{y},\mathbf{a}_y,b_y)\in\mathcal{L}$. The challenger adds the tuple $(\mathtt{Inv}(\mathfrak{y}),-\mathbf{a}_y,-b_y)$ to $\mathcal{L}$.
6. If at any point in time we have a collision, i.e., two distinct tuples $(\mathfrak{y},\mathbf{a}_1,b_1),(\mathfrak{y},\mathbf{a}_2,b_2)\in\mathcal{L}$ with $(\mathbf{a}_1,b_1)\neq(\mathbf{a}_2,b_2)$, then the event BRIDGE^N occurs, and the output of the game is 1. If BRIDGE^N never occurs, then the output of the game is 0.

Lemma 4.3 gives the upper bound of the probability of winning the BRIDGE^N-finding game for a bit-fixing preprocessing attacker. One big advantage to analyzing the multi-user bridge game *with preprocessing* in the BF-GGM is that we can avoid the complicated compression argument as shown in prior work [6, Theorem 7] and follow a similar proof technique that was used in the analysis *without preprocessing* [6, Theorem 5] which is much simpler. One notable difference is that in the online phase, there are P extra tuples $(\tau(t_1),\mathbf{0},t_1),\ldots,(\tau(t_P),\mathbf{0},t_P)$ in the list $\mathcal{L}$ when the challenger initializes it[5]. The bit-fixing attacker can only pre-fix P input/output pairs in the preprocessing phase and the hint for the online phase can only be dependent to those pairs, which means that after adding P extra tuples in the initialization of the online phase, it works identical to the bridge-finding game without preprocessing.

While we can largely utilize the same proof technique described in [6, Theorem 5], it is important to emphasize that it cannot be applied in a black-box manner. If we follow exactly the same proof technique in [6, Theorem 5] except for the modification that the initial size of the list $\mathcal{L}$ becomes $N+P+1$ (instead of $N+1$ in the prior case [6, Theorem 5]) due to P pre-fixed points, then we would obtain the following bound of the probability of winning the multi-user bridge game with preprocessing in the bit-fixing model:

$$\Pr\left[\mathsf{BridgeChal}^{\tau,N}_{\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}}(\lambda)=1\right]\le\frac{q_{\mathsf{G}}^{\mathsf{on}}(N+P)+3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}}+1)/2}{p-(N+P+3q_{\mathsf{G}}^{\mathsf{on}}+1)^2-N}. \tag{1}$$

However, an immediate concern arises as the group size p becomes larger compared to the prior work [6], resulting in longer signature lengths. Notably, there is a P^2 term in the denominator, indicating that the group size p should be at least P^2. Considering the security loss of $O\Big((S+\log\gamma^{-1})q_{\mathsf{G}}^{\mathsf{on}})/P\Big)$ [11, Theorem 1] that appears in the transition from the Bit-Fixing to the Auxiliary-Input model, we find that P must be $\Omega(2^\lambda S)$ to achieve λ bits of security. Consequently, $p=\Omega(P^2)=\Omega(2^{2\lambda}S^2)$. Hence, we come up with a longer signature length of $\log p+\lambda\ge3\lambda+2\log S$ if we use the bound in Equation (1), when compared to the prior work that achieves λ bits of security with a Schnorr signature length of $3\lambda+\log S$ bits [6].

Consequently, a more sophisticated analysis is required to maintain the same signature length as demonstrated in [6] while ensuring the same security level. Recall that the bridge event occurs if we find two distinct tuples $(\mathfrak{y}_1,\mathbf{a}_1,b_1),(\mathfrak{y}_2,\mathbf{a}_2,b_2)$ such that $\mathfrak{y}_1=\mathfrak{y}_2$ but $(\mathbf{a}_1,b_1)\neq(\mathbf{a}_2,b_2)$ which yields a non-trivial linear dependency on $\mathbf{x}$. Intuitively, we exploit the observation that if we pick two distinct tuples $(\mathfrak{y}_1,\mathbf{a}_1,b_1),(\mathfrak{y}_2,\mathbf{a}_2,b_2)$ such that $\mathbf{a}_1=\mathbf{a}_2=\mathbf{0}$ then we cannot produce the bridge event since otherwise we have a contradiction[6]. Since all the pre-fixed P tuples $(t_1,\mathbf{0},\tau(t_1)),\ldots,(t_P,\mathbf{0},\tau(t_P))$ are of the form $(\cdot,\mathbf{0},\cdot)$, we can get a tighter bound by splitting the list set $\mathcal{L}$ into two subsets $\mathcal{L}_0$ and $\mathcal{L}_1$ where $\mathcal{L}_0\coloneqq\{(\mathfrak{y},\mathbf{a},b)\in\mathcal{L}:\mathbf{a}=\mathbf{0}\}$ and $\mathcal{L}_1\coloneqq\{(\mathfrak{y},\mathbf{a},b)\in\mathcal{L}:\mathbf{a}\neq\mathbf{0}\}$ when analyzing the probability of the bridge event in the BF-GGM.

Intuitively, the proof works by considering the event $\overline{\mathsf{BRIDGE}}^N_{<i}$ that the event BRIDGE^N has not occurred until the $(i-1)^{th}$ query. Conditioning on $\overline{\mathsf{BRIDGE}}^N_{<i}$, we can essentially view $\mathbf{x}$ selected uniformly at random subject to the restriction that for any distinct tuples $(\mathfrak{y},\mathbf{a},b)$ and $(\mathfrak{y}',\mathbf{a}',b')$ we have $\mathbf{a}\cdot\mathbf{x}+b\neq\mathbf{a}'\cdot\mathbf{x}+b'$. This allows us to upper bound the probability $\Pr\left[\mathsf{BRIDGE}_i^N\middle|\overline{\mathsf{BRIDGE}}^N_{<i}\right]$ where BRIDGE_i^N denotes the event where BRIDGE^N occurs at the i^{th} query. The result follows by union bounding over $i\in[q_{\mathsf{G}}^{\mathsf{on}}]$. The full proof of Lemma 4.3 can be found in Appendix G.

[5]Here, we note that the challenger picks $(x_1,\ldots,x_N)\in\mathbb{Z}_p^N$ *after* such P pairs are fixed.

[6]In particular, suppose that $(\mathfrak{y}_1,\mathbf{0},b_1)$ and $(\mathfrak{y}_2,\mathbf{0},b_2)$ cause a bridge event. Then it should be the case that $\mathfrak{y}_1=\mathfrak{y}_2$ but $(\mathbf{0},b_1)\neq(\mathbf{0},b_2)$. But this is a contradiction because $\mathfrak{y}_1=\mathfrak{y}_2$ implies that $\mathbf{0}\cdot\mathbf{x}+b_1=\mathbf{0}\cdot\mathbf{x}+b_2$ which tells us that $b_1=b_2$, resulting that $(\mathbf{0},b_1)=(\mathbf{0},b_2)$.

Lemma 4.3. *Let $p > 2^{2\lambda}$ be a prime number and $N \in \mathbb{N}$ be a parameter. Let $\mathcal{A} \coloneqq \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ fixes P input/output pairs of the labeling map τ and $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} \coloneqq q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles. Then*

$$\Pr\left[\mathsf{BridgeChal}_{\mathcal{A}}^{\tau,N}(\lambda) = 1\right] \leq \frac{q_{\mathsf{G}}^{\mathsf{on}}(N+P) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}}+1)/2}{p - 2P(N+3q_{\mathsf{G}}^{\mathsf{on}}+1) - (N+3q_{\mathsf{G}}^{\mathsf{on}}+1)^2 - N},$$

in the GGM of prime order p, where the randomness is taken over the selection of $x_1, \dots, x_N, \tau$ as well as any random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$.

4.1 Multi-User Security of Nonzero Schnorr Signatures in the BF-ROM+GGM

Theorem 4.4. *Let $\Pi_{\backslash\{0\}} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ be a nonzero Schnorr signature scheme and $p > 2^{2\lambda}$ be a prime number. Let $N \in \mathbb{N}$ be a parameter and $\left(\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ fixes P_1 input/output pairs of a random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda_1}$ and P_2 input/output pairs of the map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $2P_1 + P_2 = P$, and the hint $\mathsf{str}_{\tau,\mathsf{H}}$ is only dependent on those P points. If $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} \coloneqq q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles, $q_{\mathsf{H}}^{\mathsf{on}}$ queries to the random oracle, and $q_{\mathsf{S}}^{\mathsf{on}}$ queries to the signing oracle, then*

$$\Pr\left[\mathsf{SigForge}_{\mathcal{A}_{\mathsf{Sig.on},\mathsf{str}_{\tau,\mathsf{H}}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)},\Pi_{\backslash\{0\}}^{\mathsf{short}}}^{\tau,\mathsf{H},N}(\lambda) = 1\right] \leq \varepsilon,$$

with

$$\varepsilon = \frac{q_{\mathsf{G}}^{\mathsf{on}}(N+P) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}}+1)/2}{p - 2P(N+3q_{\mathsf{G}}^{\mathsf{on}}+1) - (N+3q_{\mathsf{G}}^{\mathsf{on}}+1)^2 - N} + \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P)}{p} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P}{p - (N+P+3q_{\mathsf{G}}^{\mathsf{on}}+1)} + \frac{2q_{\mathsf{H}}^{\mathsf{on}}+1}{2^{\lambda_1}},$$

where the randomness is taken over the selection of τ and the random coins of $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$.

Proof Sketch. Given a bit-fixing adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$ attacking $\Pi_{\backslash\{0\}}$, we build an efficient algorithm $\mathcal{B} = (\mathcal{B}_{\mathsf{pre}}, \mathcal{B}_{\mathsf{on}})$ which tries to win the 1-out-of-N generic BRIDGE^N-finding game $\mathsf{BridgeChal}_{\mathcal{B}}^{\tau,N}(\lambda)$ in the Bit-Fixing model. Here, $\mathcal{B}_{\mathsf{pre}}$ can simply run $\mathcal{A}_{\mathsf{pre}}$ and outputs the same P_1 input/output pairs of H and P_2 input/output pairs of τ being fixed by $\mathcal{A}_{\mathsf{pre}}$. It is noteworthy to mention that for each fixed RO input/output pair $(h_i, \mathsf{H}(h_i))$ for $i \in [P_1]$, $\mathcal{A}_{\mathsf{pre}}$ can parse $h_i = I_i' \| m_i'$ where I_i' is a bitstring of length ℓ. If I_i' is a fresh ℓ-bit string (i.e., $I_i' \neq \tau(t_j)$ for all $j \in [P_2]$), then $\mathcal{A}_{\mathsf{pre}}$ picks a value for $\tau^{-1}(I_i') =: r_i'$ since the labeling map τ has not been sampled yet in the preprocessing phase. Then $\mathcal{A}_{\mathsf{pre}}$ also outputs the fixed input/output pairs of τ for those values. Then $P_1 + (P_1 + P_2) = P$ points have been fixed by $\mathcal{A}_{\mathsf{pre}}$ in total. Now the challenger initializes the list $\mathcal{L}$ that consists of tuples of the form $(\tau(y), \mathbf{a}, b) \in \mathbb{G} \times \mathbb{Z}_p^N \times \mathbb{Z}_p$ including the initial N tuples of unknowns and incorporating the information about P fixed points.

Now the online attacker $\mathcal{B}_{\mathsf{on}}$ receives this information as hint and run $\mathcal{A}_{\mathsf{on}}$ with access to the random oracle H, the generic group oracles ($\mathtt{Mult}, \mathtt{Inv}$), and the signing oracle. $\mathcal{L}$ is updated as $\mathcal{A}_{\mathsf{on}}$ makes queries to those oracles. Whenever $\mathcal{A}_{\mathsf{on}}$ submits a query m_i to the signing oracle $\mathsf{Sign}(x_j, \cdot)$, the attacker simulates the signing oracle without knowledge of the secret key x_j using the programmability of the random oracle, i.e., to sign a message m_i, we can pick s_i and e_i randomly, compute $I_i = \tau(s_i - x_j e_i)$ and see if the random oracle has previously queried at $\mathsf{H}(I_i \| m_i)$. If not, then we can program the random oracle as $\mathsf{H}(I_i \| m_i) = e_i$ and output the signature (s_i, e_i). Otherwise, the reduction simply outputs $\perp$ for failure. Since s_i and e_i are selected randomly, we can argue that the probability for outputting $\perp$ here is small, i.e., $\leq \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)}{p}$.

After $\mathcal{A}_{\mathsf{on}}$ outputs a forged signature $\sigma_{i*} = (s_{i*}, e_{i*})$ and the message m_{i*}, we compute $\tau(-e_{i*}x_{i*})$, $\mathfrak{s}_{i*} = \mathsf{Pow}(\mathfrak{g}, s_{i*})$, and $I_{i*} = \mathtt{Mult}(\mathfrak{s}_{i*}, \tau(-e_{i*}x_{i*}))$, which ensures that $(I_{i*}, -e_{i*}\hat{u}_{i*}, s_{i*}) \in \mathcal{L}$ and see if we have a bridge event. One failure event is that I_{i*} was not previously recorded in $\mathcal{L}$ so that we cannot find the bridge event. Another failure event is that I_{i*} was previously recorded in $\mathcal{L}$ but the exact same tuple $(I_{i*}, -e_{i*}\hat{u}_{i*}, s_{i*})$ was in $\mathcal{L}$. We can prove that the probability of both failure events is also small, i.e., $\leq \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P}{p - (N+P+3q_{\mathsf{G}}^{\mathsf{on}}+1)} + \frac{2q_{\mathsf{H}}^{\mathsf{on}}+1}{2^{\lambda_1}}$.

If those failure events do not occur, then BRIDGE^N occurs and $\mathcal{B}$ wins the multi-user bridge-finding game. By applying Lemma 4.3, we get the desired result (see Appendix G for the full proof). □

4.2 From Bit-Fixing Model to Auxiliary-Input Model

By applying Theorem 3.2, we can address the multi-user security of *nonzero Schnorr signatures* against preprocessing attacks in the AI-ROM+GGM.

Theorem 4.5. *Let $\Pi_{\backslash\{0\}} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ be a nonzero Schnorr signature scheme, $p > 2^{2\lambda}$ be a prime number, and $\gamma > 0$ be a parameter. Let $N \in \mathbb{N}$ be a parameter and $\left(\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{AI\text{-}RO+GG}}, \mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{AI\text{-}RO+GG}}\right)$ be a pair of generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{AI\text{-}RO+GG}}$ outputs an S-bit hint $\mathsf{str}_{\mathsf{H},\tau}$. If $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{AI\text{-}RO+GG}}$ takes $\mathsf{str}_{\mathsf{H},\tau}$ as input and makes at most $q_{\mathsf{G}}^{\mathsf{on}} \coloneqq q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles, $q_{\mathsf{H}}^{\mathsf{on}}$ queries to the random oracle, and $q_{\mathsf{S}}^{\mathsf{on}}$ queries to the signing oracle, then $\Pr\left[\mathsf{SigForge}_{\mathcal{A}_{\mathsf{Sig.on},\mathsf{str}_{\tau,\mathsf{H}}}^{\mathsf{AI\text{-}RO+GG}},\Pi_{\backslash\{0\}}^{\mathsf{short}}}^{\tau,\mathsf{H},N}(\lambda) = 1\right] \leq \varepsilon$, with*

$$\varepsilon = \frac{2q_{\mathsf{G}}^{\mathsf{on}}(N+Sq) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}}+1)}{p - 2Sq(N+3q_{\mathsf{G}}^{\mathsf{on}}+1) - (N+3q_{\mathsf{G}}^{\mathsf{on}}+1)^2 - N} + \frac{2q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + Sq)}{p} + \frac{2q_{\mathsf{H}}^{\mathsf{on}} + 2q_{\mathsf{S}}^{\mathsf{on}} + 2Sq}{p - (N+Sq+3q_{\mathsf{G}}^{\mathsf{on}}+1)} + \frac{2q_{\mathsf{H}}^{\mathsf{on}}+1}{2^{\lambda_1-1}} + 2^{-2\lambda+1},$$

where $q \coloneqq q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}}$ and the randomness is taken over the selection of τ and the random coins of $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{AI\text{-}RO+GG}}$.

Proof. This is straightforward by combining Theorem 4.4 with Theorem 3.2 with setting $P \approx Sq$ and $\gamma = 2^{-2\lambda}$. □

4.3 Discussion

Tightness of the Bound. We emphasize that the upper bound in Theorem 4.5 is tight. Under the reasonable assumption that $N \ll q$ and $6Sq^2 \ll p$, the dominating terms are $O\big(Sq^2/p\big)$ and

$O(q_{\mathsf{H}}^{\mathsf{on}}/2^{\lambda_1})$. Blocki and Lee [6] observed that a preprocessing attacker can solve one of the N discrete-log challenges with probability at least $\Omega(S(q_{\mathsf{G}}^{\mathsf{on}})^2/p)$ which enables recovery of a secret key and trivial signature forgery. This matches the $O(Sq^2/p)$ term from our bounds. Furthermore, one could also consider the following attack: pick random nonzero values of s, e then search for a message m for which (s, e) is a valid forgery, i.e., fix s, e, compute $R = g^s \cdot \mathsf{pk}^{-e}$, and search for m such that $\mathsf{H}(R\|m) = e$. This attack succeeds with probability $\Omega(q_{\mathsf{H}}^{\mathsf{on}}/2^{\lambda_1})$ showing that the $O\left(q_{\mathsf{H}}^{\mathsf{on}}/2^{\lambda_1}\right)$ term is necessary.

Practical Parameters for Fixed Implementations. Consider a fixed implementation of a nonzero Schnorr signature with an elliptic group of size $p \approx 2^{384}$, e.g., P-384. Suppose that a preprocessing attacker utilizes a hint of size $S \approx 2^{72}$. We remark that 2^{72} plausibly exceeds the storage capacity of Meta's data warehouse which was 300 petabytes in 2014[7] and has now reached the exabytes scale[8]. If an online attacker makes $\leq q \approx 2^{80}$ queries, Theorem 4.5 tells us that the success probability of forging a signature will be $\approx 2Sq^2/p \approx 2^{-151}$. Table 2 in Appendix H shows the (approximate) upper bound on the probability that a preprocessing attacker wins the 1-out-of-N signature forgery game for nonzero Schnorr signatures.

Elliptic Curve Groups for Desired Security Levels. To achieve λ bits of security for nonzero short Schnorr signatures against preprocessing attackers, we need to instantiate with bigger elliptic curve groups than against attackers without preprocessing. While NIST recommends using elliptic curve groups with size 2λ bits (i.e., $p \approx 2^{2\lambda}$) to achieve λ bits of security, one would need to fix the group size p such that $p \approx 2^{2\lambda}S$ for λ bits of security against *preprocessing attackers*. For example, to achieve 192 bits of security, NIST recommends to use elliptic curves of size ≥ 384 bits; for preprocessing security against an attacker with an $S \leq 2^{64}$-bit hint, we suggest instantiate with an elliptic curve group of size ≥ 448 bits, e.g., use W-448, Curve448, Edwards448, E448, or P-521 (using a prime field $\mathbb{F}_{\mathfrak{p}}$ with prime $\mathfrak{p}$ of size 521 bits), based on the NIST SP 800-186 [8]. If we further aim to achieve 224 bits of *preprocessing* security against an attacker with a $S \leq 2^{72}$-bit hint, we recommend using an elliptic curve of size ≥ 520 bits, such as P-521. If we aim to achieve 256 bit bits of *preprocessing preprocessing* security against an attacker with a $S \leq 2^{80}$-bit hint, then it will be necessary to standardize new elliptic curve groups with larger sizes. Table 1 provides a summary of suggested elliptic curve group sizes for various NIST security levels to ensure preprocessing security for nonzero Schnorr signatures.

Signature Length for Nonzero Short Schnorr Signatures. To achieve λ bits of multi-user security for short Schnorr signatures with preprocessing, we can fix p such that $p \approx 2^{2\lambda}S$, and set the length of our hash output to be $\lambda_1 = \lambda + 2$. With these parameters, Theorem 4.5 tells us that a preprocessing attacker *in the AI-ROM+GGM* wins the signature forgery game with probability at most $\varepsilon = O(q/2^{\lambda})$. The length of the signatures we obtain will be $\lambda+2+\log p = 3\lambda+2+\log S$.

NIST Security Level	Current ECG	Suggested ECG for Preprocessing Security		
(bits)	(min. key size)	$S = 2^{64}$	$S = 2^{72}$	$S = 2^{80}$
112	$224 \leq$	$288 \leq$	$296 \leq$	$304 \leq$
128	$256 \leq$	$320 \leq$	$328 \leq$	$336 \leq$
192	$384 \leq$	$448 \leq$	$456 \leq$	$464 \leq$
224	$448 \leq$	$512 \leq$	$520 \leq$	$528 \leq$
256	$512 \leq$	$576 \leq$	$584 \leq$	$592 \leq$

Table 1: Suggested minimum elliptic curve group (ECG) sizes for nonzero Schnorr signatures to achieve various NIST security levels against preprocessing attacks.

As a concrete example, if $S \leq 2^{9\lambda/16}$, then we obtain signatures of length $\approx 3.5625\lambda$ while regular Schnorr signatures are 4λ bits. If we want to have $\lambda \geq 128$ bits of security, then the assumption that $S \leq 2^{9\lambda/16}$ seems quite reasonable since 2^{72} bits exceeds the current storage capacity of Meta's data warehouse. As a second example, if we take $S \leq 2^{80}$ as an even more conservative upper bound on the storage capacity of any nation-state attacker, then we obtain signatures of length $\approx 3\lambda + 80$.

5 Security of PSEC-KEM with Preprocessing

As the second application, we analyze the preprocessing security of a key encapsulation mechanism called PSEC-KEM (Provably Secure Elliptic Curve-Key Encapsulation Mechanism). PSEC-KEM [41, 43] is an El-Gamal-based key encapsulation mechanism first proposed by NTT. PSEC-KEM has several standardized implementation such as NESSIE [41] and ISO/IEC 18033-2 [22]. PSEC-KEM was also certified by IETF (Internet Engineering Task Force) for the XML security of URIs [1]. PSEC-KEM has provable security in the ROM and under the hardness assumption of the elliptic curve discrete logarithm [7, 32, 41], but the preprocessing security of PSEC-KEM has never been established. To the best of our knowledge, our work is the first to prove *preprocessing security* of PSEC-KEM in the ROM+GGM. We first formally define the PSEC-KEM scheme in the ROM+GGM below. Here, we stress that in the scheme, three separate random oracles $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{\lambda}$, and $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{\lambda_1}$ (where λ_1 is a parameter which will be set later) are used that is consistent with the PSEC-KEM specification.

The PSEC-KEM Scheme.

Consider the GGM with labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ for a prime p and the set of bitstrings $\mathbb{G} = \{0,1\}^{\lceil \log p \rceil}$. The PSEC-KEM scheme consists of three algorithms (Gen, Encaps, Decaps), where each algorithm works as follows:

- $\mathsf{Gen}(1^{\lambda})$: the key-generation algorithm taking as input the security parameter 1^{λ} and outputs a public/secret-key pair $(\mathsf{pk}, \mathsf{sk}) \coloneqq (\tau(x), x)$ for a random $x \in \mathbb{Z}_p$.
- $\mathsf{Encaps}(\mathsf{pk}, 1^{\lambda})$: the encapsulation algorithm taking as input $\mathfrak{g} = \tau(1)$, $\mathsf{pk} = \tau(x)$, and 1^{λ} and outputs a key k and a ciphertext c as follows:
    - Pick a random $r \in \{0,1\}^{\lambda_1}$.
    - Compute $\mathsf{H}_0(r) = \alpha$ and $\mathsf{H}_1(r) = k$.

[7]See the link: https://engineering.fb.com/2014/04/10/core-data/scaling-the-facebook-data-warehouse-to-300-pb/
[8]See the link: https://medium.com/@AnalyticsAtMeta/data-engineering-at-meta-high-level-overview-of-the-internal-tech-stack-a200460a44fe

  - Compute $c_1 = r \oplus \mathsf{H}_2(\tau(\alpha)\|\tau(\alpha x))$ and $c := (\tau(\alpha), c_1)$.
    (Note: $\tau(\alpha) \leftarrow \mathsf{Pow}(\mathfrak{g}, \alpha)$ and $\tau(\alpha x) \leftarrow \mathsf{Pow}(\mathsf{pk}, \alpha)$)
  - Output a key-ciphertext pair (k, c). We write $k \leftarrow \mathsf{Encaps}(\mathsf{pk}, 1^\lambda).\mathsf{key}$ and $c \leftarrow \mathsf{Encaps}(\mathsf{pk}, 1^\lambda).\mathsf{ctxt}$.
- $\mathsf{Decaps}(\mathsf{sk}, c)$: the deterministic decapsulation algorithm taking as input a secret key $\mathsf{sk} = x$ and a ciphertext c, and outputs a key k or $\perp$ denoting failure as follows:
  - Parse $c = (\mathfrak{h}, c_1)$.
  - Compute $r = c_1 \oplus \mathsf{H}_2(\mathfrak{h}\|\mathsf{Pow}(\mathfrak{h}, x))$.
  - Compute $\mathsf{H}_0(r) = \alpha$ and $\mathsf{H}_1(r) = k$.
  - Verify that $\mathfrak{h} = \tau(\alpha)$. If verified, output k; otherwise, output $\perp$.

We then define the CPA (Chosen Plaintext Attack) security of a KEM $\Pi = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ in the ROM+GGM via the CPA indistinguishability game $\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda)$. Intuitively, the game works as follows: the attacker is given the public key pk and a key-ciphertext pair $(\hat{k}, c)$. Ultimately, the attacker is asked to distinguish whether $\hat{k}$ is the challenger's real encapsulated key $k \leftarrow \mathsf{Encaps}(\mathsf{pk}, 1^\lambda).\mathsf{key}$ or a randomly selected string. The challenger picks a uniform bit $b \in \{0, 1\}$ and selects the real encapsulated key if $b = 0$ and selects a random key if $b = 1$. The attacker is given access to the "encapsulation" oracle $\mathsf{Encaps}_b(\cdot)$, where $\mathsf{Encaps}_b(\mathsf{pk})$ first runs $(k, c) \leftarrow \mathsf{Encaps}(\mathsf{pk}, 1^\lambda)$ and then outputs (k_b, c) where $k_0 = k$ and $k_1 \leftarrow\$ \{0, 1\}^\lambda$ is a uniformly random key unrelated to the ciphertext c. The attacker is also given access to the random oracles and the generic group oracle $(\mathtt{Mult}(\cdot, \cdot), \mathtt{Inv}(\cdot))$. After multiple queries to the oracles above, the attacker finally outputs a bit b', and wins the security game if $b' = b$. The KEM is said to be CPA-secure if distinguishing between the two is not more efficient than a random guess, i.e., the distinguishing probability is not greater than $1/2 + \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\lambda)$. See Appendix E.1 for the formal description of the game $\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda)$.

Definition 5.1. Consider the generic group model with the labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$. A KEM $\Pi = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ is said to be $(q_\mathsf{H}, q_\mathsf{G}, q_\mathsf{E}, \varepsilon)$*-CPA secure (secure against chosen-plaintext attack)* if for every adversary $\mathcal{A}$ making at most q_H, q_G, and q_E queries to the random oracles, generic group oracles, and $\mathsf{Encaps}_b(\cdot)$, respectively, the following bound holds:

$$\Pr\left[\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \le \frac{1}{2} + \varepsilon,$$

where the randomness is taken over the selection of τ, the random coins of $\mathcal{A}$, the random coins of Gen, the selection of random oracles $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$, and the random coins of Encaps_b.

When it comes to the *preprocessing security*, we say that a KEM Π is CPA-secure against preprocessing attacks if for every adversary $\mathcal{A} = (\mathcal{A}_\mathsf{pre}, \mathcal{A}_\mathsf{on})$, Π is $(q_\mathsf{H}, q_\mathsf{G}, q_\mathsf{E}, \varepsilon)$-CPA secure against the *online* attacker $\mathcal{A}_\mathsf{on}$. In the Bit-Fixing model, $\mathcal{A}_\mathsf{on}$ will receive the information of P pre-fixed points from $\mathcal{A}_\mathsf{pre}$, and in the Auxiliary-Input model, $\mathcal{A}_\mathsf{on}$ will receive an S-bit hint from $\mathcal{A}_\mathsf{pre}$. Hence, to prove the preprocessing security, it is sufficient to upper bound the probability of the event $\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A}_\mathsf{on},\Pi}(\lambda) = 1$ for $\mathcal{A}_\mathsf{on}$.

As a warmup, we first analyze the CPA security of PSEC-KEM in the ROM+GGM without preprocessing in Appendix D. For the rest of the paper, we will focus on the *preprocessing security* of PSEC-KEM in the ROM+GGM.

5.1 The CPA Security of PSEC-KEM in the BF-ROM+GGM

We first establish the CPA *preprocessing* security of PSEC-KEM in the BF-ROM+GGM via reduction from a new game called the *quadratic bridge-finding game.*

The Quadratic Bridge-Finding Game in the BF-GGM. We define the *quadratic bridge-finding game* $\mathsf{QDBridgeChal}^\tau_\mathcal{A}(\lambda)$ in the BF-GGM. The game is run by a challenger and an attacker $\mathcal{A} = (\mathcal{A}_\mathsf{pre}, \mathcal{A}_\mathsf{on})$. In a preprocessing phase, the preprocessing attacker fixes P input/output pairs of a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$, i.e., $(t_1, \tau(t_1)), \ldots, (t_P, \tau(t_P))$. Then the rest of the map τ is chosen uniformly at random subject to the constraint that the map remains injective. In an online phase, the challenger receives P pre-fixed points, picks a secret $x \leftarrow\$ \mathbb{Z}_p$, and sends $\tau(1)$ and $\tau(x)$ to the online attacker $\mathcal{A}_\mathsf{on}$. $\mathcal{A}_\mathsf{on}$ is also given P pre-fixed points of the map τ as a hint from $\mathcal{A}_\mathsf{pre}$, and has access to the generic group oracle $(\mathtt{Mult}(\cdot, \cdot), \mathtt{Inv}(\cdot))$ and a generic oracle O which selects a new random unknown variable $x_j \leftarrow\$ \mathbb{Z}_p$ and outputs $\tau(x_j)$.

The attacker's goal is to find a *quadratic relationship* between the unknown variables; it is not required for the attacker to solve for those unknown variables. In particular, the attacker needs to output a tuple $(i, j, \mathbf{a}, b)$ which satisfies $x_i x_j = \mathbf{a} \cdot \mathbf{x} + b$ to win the game, denoted by $\mathsf{QDBridgeChal}^\tau_\mathcal{A}(\lambda) = 1$, where the first two elements in the tuple denote the indices of the unknown variables that create a quadratic term, and the vector $\mathbf{a}$ and the scalar b create the linear relationship between the unknowns.[9]

There are two important things to note. (1) As the attacker makes query to the oracles, the challenger maintains the list $\mathcal{L}$ that consists of tuples of the form $(\mathfrak{y}, \mathbf{a}, b)$ which indicates the relationship $\mathfrak{y} = \tau(\mathbf{a} \cdot \mathbf{x} + b)$ where $\mathbf{x}$ is the vector of unknowns. Initially, $\mathcal{L} = \{(\tau(1), 0, 1), (\tau(x), 1, 0), (\tau(t_1), 0, t_1), \ldots, (\tau(t_P), 0, t_P)\}$ since there is only one unknown variable x and the challenger has the information of P pre-fixed points. (2) We allow the unknown variables to *expand* as the game progresses. There are several occasions when this happens (suppose $\dim(\mathbf{x}) = j$). A few examples are:

- When the attacker makes a query to the oracle O, then the oracle picks a new random unknown $x_{j+1} \leftarrow\$ \mathbb{Z}_p$ and outputs $\tau(x_{j+1})$. The challenger updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1}$ and updates all the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$ as the number of unknowns is increased by 1. Finally, the challenger adds $(\tau(x_{j+1}), \mathbf{0} \circ 1, 0)$ to the list $\mathcal{L}$.
- When the attacker queries $\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2)$, and if $(\mathfrak{y}_1, \mathbf{a}_1, b_1) \in \mathcal{L}$ but $\mathfrak{y}_2$ does not appear in $\mathcal{L}$, then the challenger updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1}$ where x_{j+1} denotes a new unknown satisfying $\tau(x_{j+1}) = \mathfrak{y}_2$. Similarly, the challenger also updates all the

[9] For example, suppose there are 3 unknown variables x_1, x_2, x_3. The attacker wins the quadratic bridge-finding game if s/he outputs a tuple $(2, 3, (5, -3, 2), -1)$, which indicates the quadratic relationship $x_2 x_3 = (5, -3, 2) \cdot (x_1, x_2, x_3) - 1 = 5x_1 - 3x_2 + 2x_3 - 1$. Note that we can always make the coefficient of the quadratic term 1 as p is prime, e.g., if the attacker ever found the quadratic relationship $c x_i x_j = \mathbf{a} \cdot \mathbf{x} + b$ for $c \neq 0$ then s/he can output $(i, j, c^{-1}\mathbf{a}, c^{-1}b)$ since c^{-1} always exists in $\mathbb{Z}_p$ if $c \neq 0$.

entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$, and adds $(\mathfrak{y}_2, \mathbf{0} \circ 1, 0)$ and $(\texttt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{a}_1 \circ 0 + \mathbf{0} \circ 1, b_1)$ to $\mathcal{L}$.

See Appendix E.2 for the formal description of the quadratic bridge-finding game, which handles all the possible cases when the attacker queries $\texttt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2)$ or $\texttt{Inv}(\mathfrak{y})$.

Lemma 5.2 upper bounds the probability of winning the quadratic bridge-finding game for a bit-fixing preprocessing attacker. The proof works largely the same as the proof of Lemma D.1 in Appendix D, which upper bounds the odds of winning the quadratic bridge-finding game without preprocessing. Intuitively, the proof works by maintaining a list $\mathcal{L}$ of tuples $(\tau(y), \mathbf{a}, b) \in \mathbb{G} \times \mathbb{Z}_p^{\dim(\mathbf{x})} \times \mathbb{Z}_p$ which satisfies the constraint $y = \mathbf{a} \cdot \mathbf{x} + b$. We consider the event BRIDGE which denotes the event that the attacker finds some linear dependency on $\mathbf{x}$. Conditioning on $\overline{\mathsf{BRIDGE}}$, we can essentially view $\mathbf{x}$ selected uniformly at random subject to the restriction that for any distinct tuples $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$ we have $\mathbf{a} \cdot \mathbf{x} + b \neq \mathbf{a}' \cdot \mathbf{x} + b'$. For $x_r = \mathbf{x}[r]$ such that $\mathbf{a}[r] - \mathbf{a}'[r] \neq 0$, if the attacker outputs a tuple $(i_1, i_2, \mathbf{a}, b)$ such that $x_{i_1} x_{i_2} = \mathbf{a} \cdot \mathbf{x} + b$ then there are at most 2 roots in $\mathbb{Z}_p$ since this is a quadratic equation at the worst. This allows us to upper bound the probability $\Pr[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1 | \overline{\mathsf{BRIDGE}}]$. The analysis of $\Pr[\mathsf{BRIDGE}]$ follows a similar approach to the proof of Lemma 4.3. Now we obtain our result by upper bounding the probability $\Pr[\mathsf{BRIDGE}] + \Pr[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1 | \overline{\mathsf{BRIDGE}}]$ which upper bounds the desired probability $\Pr[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1]$. The full proof of Lemma 5.2 can be found in Appendix G.

Lemma 5.2. *Let $p > 2^{2\lambda}$ is a prime number. Let $\mathcal{A} := \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ fixes P input/output pairs of the labeling map τ and $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} := q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles and $q_{O}^{\mathsf{on}} := q_{O}^{\mathsf{on}}(\lambda)$ queries to the oracle O. Then $\Pr\left[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1\right] \leq \varepsilon$, where*

$$\varepsilon := \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})^2 + (5 + 2P)(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}}) + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 - 2(2q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})},$$

in the GGM of prime order p, where the randomness is taken over the selection of $x_1, \ldots, x_N, \tau$ as well as any random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$.

We are now ready to prove the CPA security of PSEC-KEM against preprocessing attacks in the BF-ROM+GGM.

Theorem 5.3. *Let $\Pi = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ be PSEC-KEM and $p > 2^{2\lambda}$ be a prime number. Let $\mathcal{A} = \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ fixes $P_{1,1}$ (resp. $P_{1,2}, P_{1,3}$) input/output pairs of a random oracle $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_p$ (resp. $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{\lambda}$, $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{\lambda_1}$), and P_2 input/output pairs of the map τ such that $P_{1,1} + P_{1,2} + P_{1,3} = P_1$ and $2P_1 + P_2 = P$, and the hint $\mathsf{str}_{\tau,\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2}$ is only dependent on those P points. If $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ makes at most $q_{\mathsf{H}}^{\mathsf{on}}$ (resp. $q_{\mathsf{G}}^{\mathsf{on}}, q_{\mathsf{E}}^{\mathsf{on}}$) queries to the random oracle (resp. generic group oracle, encapsulation oracle), then $\Pr\left[\mathsf{KEM}^{\tau,(\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2),\mathsf{cpa}}_{\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{on},\mathsf{str}_{\tau,\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2}},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \varepsilon$, with*

$$\varepsilon = \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}})^2 + (5 + 2P)(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}}) + 4}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1)^2 - 2(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{E}}^{\mathsf{on}})} + \frac{3(q_{\mathsf{H}}^{\mathsf{on}} + P)q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}} + \frac{(q_{\mathsf{H}}^{\mathsf{on}} + P)q_{\mathsf{E}}^{\mathsf{on}}}{p},$$

where the randomness is taken over the selection of τ and the random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$.

Proof Sketch. We use a hybrid argument. In the first hybrid (hybrid H_0), the distinguisher $\mathcal{D}$ is given the CPA indistinguishability game for PSEC-KEM with the challenge bit $b = 0$, and in the last hybrid (hybrid H_3), $\mathcal{D}$ is given the CPA indistinguishability game with $b = 1$. The intermediate hybrids are defined as follows. Hybrid H_1 is the same as H_0 except that the encapsulation oracle is modified so that $\tau(\alpha)$ and $\tau(\alpha x)$ are replaced with random elements in $\tau(\mathbb{Z}_p)$ by querying the oracle O twice. Hybrid H_2 is the same as H_1 except that the encapsulation oracle is further modified so that the key k is sampled uniformly at random from $\{0,1\}^{\lambda}$.

We show that H_0, H_1 (resp. H_2, H_3) are perfectly indistinguishable unless (1) the random oracle query $\mathsf{H}_2(\tau(\alpha) \| \tau(\alpha x))$ has been made, or (2) the adversary makes query r to H_0 (which is not one of the fixed points of H_0) but has not queried $\mathsf{H}_2(\tau(\alpha) \| \tau(\alpha x))$, or (3) r is one of the fixed points of H_0. We show that Case (1) reduces to winning the quadratic bridge-finding game so that we can apply Lemma 5.2 with $q_{O}^{\mathsf{on}} \leq 2q_{\mathsf{E}}^{\mathsf{on}}$. Since there are at most P fixed points, by union bound we can show that Case (2) has probability $\leq \frac{q_{\mathsf{H}}^{\mathsf{on}} \cdot q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}}$ and Case (3) has probability $\leq \frac{P q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}}$. We also prove that the distinguishing probability between H_1, H_2 is at most $q_{\mathsf{E}}^{\mathsf{on}} \left(\frac{q_{\mathsf{H}}^{\mathsf{on}} + P}{p} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + P}{2^{\lambda_1}}\right)$ since two hybrids are perfectly indistinguishable unless $\mathsf{H}_1(r)$ is queried (which is not one of the fixed points) or r is one of the fixed points. Taken together, we can conclude the proof. The full proof can be found in Appendix G. □

5.2 The CPA Security of PSEC-KEM in the AI-ROM+GGM

By applying Theorem 3.2, we can address the CPA security of PSEC-KEM against preprocessing attacks in the AI-ROM+GGM.

Theorem 5.4. *Let $\Pi = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ be PSEC-KEM and $p > 2^{2\lambda}$ be a prime number. Let $\mathcal{A} = \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{AI\text{-}RO+GG}}, \mathcal{A}_{\mathsf{on}}^{\mathsf{AI\text{-}RO+GG}}\right)$ be a pair of (auxiliary-input) generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{AI\text{-}RO+GG}}$ outputs an S-bit hint $\mathsf{str}_{\tau,\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2}$. If $\mathcal{A}_{\mathsf{on}}^{\mathsf{AI\text{-}RO+GG}}$ makes at most $q_{\mathsf{H}}^{\mathsf{on}}$ (resp. $q_{\mathsf{G}}^{\mathsf{on}}, q_{\mathsf{E}}^{\mathsf{on}}$) queries to the random oracles $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{\lambda}$ and $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{\lambda_1}$ in total (resp. generic group oracle, encapsulation oracle), then $\Pr\left[\mathsf{KEM}^{\tau,(\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2)\mathsf{cpa}}_{\mathcal{A}^{\mathsf{AI\text{-}RO+GG}}_{\mathsf{on},\mathsf{str}_{\tau,\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2}},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \varepsilon$, with*

$$\varepsilon = \frac{6(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}})^2 + 2(5 + 2Sq)(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}}) + 8}{p - 2Sq(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1)^2 - 2(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{E}}^{\mathsf{on}})} + \frac{6(q_{\mathsf{H}}^{\mathsf{on}} + Sq)q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}} + \frac{2(q_{\mathsf{H}}^{\mathsf{on}} + Sq)q_{\mathsf{E}}^{\mathsf{on}}}{p} + 2^{-2\lambda+1},$$

where $q \coloneqq q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{E}}^{\mathsf{on}}$ denotes the total number of online *queries made by a preprocessing attacker and the randomness is taken over the selection of τ and the random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{AI\text{-}RO+GG}}$.*

Proof. This is straightforward by combining Theorem 5.3 with Theorem 3.2 with setting $P = Sq$ and $\gamma = 2^{-2\lambda}$. □

Instantiation with Parameters. We would like to have the distinguishing probability in Theorem 5.4 bounded by $(1/2) + O\left(q/2^{\lambda}\right)$ for any $q \le 2^{\lambda}$. To achieve λ bits of security for PSEC-KEM with preprocessing, we can fix p such that $p \approx 2^{2\lambda}S$ and set the length of the output of H_2 to be $\lambda_1 = 2\lambda + \log_2(6S)$. Then Theorem 5.4 tells us that a preprocessing attacker *in the AI-ROM+GGM* wins the CPA indistinguishability game with probability at most $(1/2) + O\left(q/2^{\lambda}\right)$.

References

[1] Donald E. Eastlake 3rd. 2005. Additional XML Security Uniform Resource Identifiers (URIs). RFC 4051. https://doi.org/10.17487/RFC4051

[2] Akshima, David Cash, Andrew Drucker, and Hoeteck Wee. 2020. Time-Space Tradeoffs and Short Collisions in Merkle-Damgård Hash Functions. In *CRYPTO 2020, Part I (LNCS, Vol. 12170)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Cham, 157–186. https://doi.org/10.1007/978-3-030-56784-2_6

[3] Akshima, Siyao Guo, and Qipeng Liu. 2022. Time-Space Lower Bounds for Finding Collisions in Merkle-Damgård Hash Functions. In *CRYPTO 2022, Part III (LNCS, Vol. 13509)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, Cham, 192–221. https://doi.org/10.1007/978-3-031-15982-4_7

[4] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM CCS 93*, Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby (Eds.). ACM Press, 62–73. https://doi.org/10.1145/168588.168596

[5] Daniel J. Bernstein and Tanja Lange. 2013. Non-uniform Cracks in the Concrete: The Power of Free Precomputation. In *ASIACRYPT 2013, Part II (LNCS, Vol. 8270)*, Kazue Sako and Palash Sarkar (Eds.). Springer, Berlin, Heidelberg, 321–340. https://doi.org/10.1007/978-3-642-42045-0_17

[6] Jeremiah Blocki and Seunghoon Lee. 2022. On the Multi-user Security of Short Schnorr Signatures with Preprocessing. In *EUROCRYPT 2022, Part II (LNCS, Vol. 13276)*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, Cham, 614–643. https://doi.org/10.1007/978-3-031-07085-3_21

[7] D. Galindo Chacon, Sébastien Martin, and Jorge Luis Villar. 2005. The security of PSEC-KEM versus ECIES-KEM. https://api.semanticscholar.org/CorpusID:15798138

[8] Lily Chen, Dustin Moody, Karen Randall, Andrew Regenscheid, and Angela Robinson. 2023. Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters. https://doi.org/10.6028/NIST.SP.800-186

[9] Jung Hee Cheon. 2010. Discrete Logarithm Problems with Auxiliary Inputs. *Journal of Cryptology* 23, 3 (July 2010), 457–476. https://doi.org/10.1007/s00145-009-9047-0

[10] Tung Chou and Claudio Orlandi. 2015. The Simplest Protocol for Oblivious Transfer. In *Proceedings of the 4th International Conference on Progress in Cryptology – LATINCRYPT 2015 - Volume 9230*. Springer-Verlag, Berlin, Heidelberg, 40–58. https://doi.org/10.1007/978-3-319-22174-8_3

[11] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. 2018. Non-Uniform Bounds in the Random-Permutation, Ideal-Cipher, and Generic-Group Models. In *CRYPTO 2018, Part I (LNCS, Vol. 10991)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Cham, 693–721. https://doi.org/10.1007/978-3-319-96884-1_23

[12] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. 2018. Random Oracles and Non-uniformity. In *EUROCRYPT 2018, Part I (LNCS, Vol. 10820)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer, Cham, 227–258. https://doi.org/10.1007/978-3-319-78381-9_9

[13] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. 2008. The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In *CRYPTO 2008 (LNCS, Vol. 5157)*, David Wagner (Ed.). Springer, Berlin, Heidelberg, 1–20. https://doi.org/10.1007/978-3-540-85174-5_1

[14] Henry Corrigan-Gibbs and Dmitry Kogan. 2018. The Discrete-Logarithm Problem with Preprocessing. In *EUROCRYPT 2018, Part II (LNCS, Vol. 10821)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer, Cham, 415–447. https://doi.org/10.1007/978-3-319-78375-8_14

[15] Henry Corrigan-Gibbs and Dmitry Kogan. 2019. The Function-Inversion Problem: Barriers and Opportunities. In *TCC 2019, Part I (LNCS, Vol. 11891)*, Dennis Hofheinz and Alon Rosen (Eds.). Springer, Cham, 393–421. https://doi.org/10.1007/978-3-030-36030-6_16

[16] Yuanxi Dai and John P. Steinberger. 2016. Indifferentiability of 8-Round Feistel Networks. In *CRYPTO 2016, Part I (LNCS, Vol. 9814)*, Matthew Robshaw and Jonathan Katz (Eds.). Springer, Berlin, Heidelberg, 95–120. https://doi.org/10.1007/978-3-662-53018-4_4

[17] Anindya De, Luca Trevisan, and Madhur Tulsiani. 2010. Time Space Tradeoffs for Attacks against One-Way Functions and PRGs. In *CRYPTO 2010 (LNCS, Vol. 6223)*, Tal Rabin (Ed.). Springer, Berlin, Heidelberg, 649–665. https://doi.org/10.1007/978-3-642-14623-7_35

[18] Peter de Rooij. 1997. On Schnorr's Preprocessing for Digital Signature Schemes. *J. Cryptol.* 10, 1 (dec 1997), 1–16. https://doi.org/10.1007/s001459900016

[19] Morris Dworkin. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. https://doi.org/10.6028/NIST.FIPS.202

[20] Shimon Even and Yishay Mansour. 1993. A Construction of a Cipher From a Single Pseudorandom Permutation. In *ASIACRYPT'91 (LNCS, Vol. 739)*, Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto (Eds.). Springer, Berlin, Heidelberg, 210–224. https://doi.org/10.1007/3-540-57332-1_17

[21] Federal Office for Information Security. 2018. Elliptic Curve Cryptography, Version 2.1. *Technical Guideline BSI TR-03111* (Jun 2018). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_V-2-1_pdf.pdf?__blob=publicationFile&v=1

[22] International Organization for Standardization and International Electrotechnical Commission. 2006. IT Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers. *ISO/IEC 18033-2* (2006).

[23] International Organization for Standardization and International Electrotechnical Commission. 2018. IT Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms. *ISO/IEC 14888-3* (Nov 2018). https://www.iso.org/standard/76382.html

[24] Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. 2022. Time-Space Tradeoffs for Sponge Hashing: Attacks and Limitations for Short Collisions. In *CRYPTO 2022, Part III (LNCS, Vol. 13509)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, Cham, 131–160. https://doi.org/10.1007/978-3-031-15982-4_5

[25] Ashrujit Ghoshal and Ilan Komargodski. 2022. On Time-Space Tradeoffs for Bounded-Length Collisions in Merkle-Damgård Hashing. In *CRYPTO 2022, Part III (LNCS, Vol. 13509)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, Cham, 161–191. https://doi.org/10.1007/978-3-031-15982-4_6

[26] Ashrujit Ghoshal and Stefano Tessaro. 2023. The Query-Complexity of Preprocessing Attacks. In *CRYPTO 2023, Part II (LNCS, Vol. 14082)*, Helena Handschuh and Anna Lysyanskaya (Eds.). Springer, Cham, 482–513. https://doi.org/10.1007/978-3-031-38545-2_16

[27] M. Hellman. 1980. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory* 26, 4 (1980), 401–406. https://doi.org/10.1109/TIT.1980.1056220

[28] Pavel Hubáček, Ľubica Jančová, and Veronika Králová. 2022. On The Distributed Discrete Logarithm Problem with Preprocessing. Cryptology ePrint Archive, Report 2022/521. https://eprint.iacr.org/2022/521

[29] Eike Kiltz, Daniel Masny, and Jiaxin Pan. 2016. Optimal Security Proofs for Signatures from Identification Schemes. In *CRYPTO 2016, Part II (LNCS, Vol. 9815)*, Matthew Robshaw and Jonathan Katz (Eds.). Springer, Berlin, Heidelberg, 33–61. https://doi.org/10.1007/978-3-662-53008-5_2

[30] Hyung Tae Lee, Jung Hee Cheon, and Jin Hong. 2011. Accelerating ID-based Encryption based on Trapdoor DL using Pre-computation. Cryptology ePrint Archive, Report 2011/187. https://eprint.iacr.org/2011/187

[31] Ueli Maurer. 2005. Abstract Models of Computation in Cryptography. In *Proceedings of the 10th International Conference on Cryptography and Coding* (Cirencester, UK) *(IMA'05)*. Springer-Verlag, Berlin, Heidelberg, 1–12. https://doi.org/10.1007/11586821_1

[32] A. Menezes. 2001. Evaluation of security level of cryptography: The revised version of PSEC-2 (PSEC-KEM). Technical report, CRYPTREC. http://www.shiba.tao.go.jp/kenkyu/CRYPTREC/fy15/cryptrec20030424outrep.html

[33] V. I. Nechaev. 1994. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Math Notes* 55 (1994), 165. https://doi.org/10.1007/BF02113297

[34] Gregory Neven, Nigel Smart, and Bogdan Warinschi. 2009. Hash function requirements for Schnorr signatures. *Journal of Mathematical Cryptology* 3 (05 2009). https://doi.org/10.1515/JMC.2009.004

[35] National Institute of Standards and Technology (NIST). 2014. FIPS 202. sha-3 standard: Permutation-based hash and extendable-output functions. Technical report, US Department of Commerce.

[36] National Institute of Standards and Technology. 2001. Advanced Encryption Standard. *NIST FIPS PUB 197* (2001).

[37] Lior Rotem and Gil Segev. 2022. A Fully-Constructive Discrete-Logarithm Preprocessing Algorithm with an Optimal Time-Space Tradeoff. Cryptology ePrint Archive, Report 2022/583. https://eprint.iacr.org/2022/583

[38] Claus-Peter Schnorr. 1990. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89 (LNCS, Vol. 435)*, Gilles Brassard (Ed.). Springer, New York, 239–252. https://doi.org/10.1007/0-387-34805-0_22

[39] Claus-Peter Schnorr and Markus Jakobsson. 2000. Security of Signed ElGamal Encryption. In *ASIACRYPT 2000 (LNCS, Vol. 1976)*, Tatsuaki Okamoto (Ed.). Springer, Berlin, Heidelberg, 73–89. https://doi.org/10.1007/3-540-44448-3_7

[40] Claude E. Shannon. 1949. Communication theory of secrecy systems. *Bell Systems Technical Journal* 28, 4 (1949), 656–715.

[41] R. Shipsey. 2001. PSEC-KEM. Technical Report `NES/DOC/EHU/WP5/001/a`, NESSIE.

[42] Victor Shoup. 1997. Lower Bounds for Discrete Logarithms and Related Problems. In *EUROCRYPT'97 (LNCS, Vol. 1233)*, Walter Fumy (Ed.). Springer, Berlin, Heidelberg, 256–266. https://doi.org/10.1007/3-540-69053-0_18

[43] Victor Shoup. 2001. A Proposal for an ISO Standard for Public Key Encryption. Cryptology ePrint Archive, Report 2001/112. https://eprint.iacr.org/2001/112

[44] Dominique Unruh. 2007. Random Oracles and Auxiliary Input. In *CRYPTO 2007 (LNCS, Vol. 4622)*, Alfred Menezes (Ed.). Springer, Berlin, Heidelberg, 205–223. https://doi.org/10.1007/978-3-540-74143-5_12

[45] A. C.-C. Yao. 1990. Coherent Functions and Program Checkers. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing* (Baltimore, Maryland, USA) *(STOC '90)*. Association for Computing Machinery, New York, NY, USA, 84–94. https://doi.org/10.1145/100216.100226

[46] Mark Zhandry. 2022. To Label, or Not To Label (in Generic Groups). In *CRYPTO 2022, Part III (LNCS, Vol. 13509)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, Cham, 66–96. https://doi.org/10.1007/978-3-031-15982-4_3

A Formal Definition of Bit-Fixing/Auxiliary-Input ROM+GGM

A.1 Replacing Auxiliary Information by Bit-Fixing in the ROM+GGM

We first define an (H, τ)-source for a random oracle H and a generic-group random encoding map τ as follows.

Definition A.1. An (H, τ)*-source* is a pair of random variables (X, Y) where X corresponds to the function table of a random oracle $\mathsf{H} : \{0,1\}^m \to \{0,1\}^\lambda$, with range $(\{0,1\}^\lambda)^{2^m}$, and Y corresponds to the function table of an injection $\tau : \mathbb{Z}_p \to \mathbb{G}$. A source (X, Y) is called

- $(P_1, P_2, 1-\delta)$*-dense* if for $(P_1, P_2) \in \mathbb{Z}_{2^m} \times \mathbb{Z}_p$ if H is fixed on at most P_1 coordinates, τ is fixed on at most P_2 coordinates, and if for all pairs of subsets $(I_1, I_2) \subseteq \{0,1\}^m \times \mathbb{Z}_p$ of non-fixed coordinates,
$$H_\infty(X_{I_1}, Y_{I_2}) \geq (1-\delta)\left[|I_1|\lambda + \log(p - P_2)^{\underline{|I_2|}}\right],$$
where $a^{\underline{b}} \coloneqq a!/(a-b)!$ and X_I is X restricted to the coordinates in I.
- $(1-\delta)$*-dense* if it is $(0, 0, 1-\delta)$-dense, and
- (P_1, P_2)*-Bit-Fixing* if it is fixed on at most P_1 coordinates on H and P_2 coordinates on τ and uniform on the rest.

Lemma A.2. *Let (X_1, X_2) be a pair of random variables such that X_1 is distributed uniformly over $(\{0,1\}^\lambda)^{2^m}$ and X_2 is a uniformly random injection that takes on as value function tables corresponding to an injection $\tau : \mathbb{Z}_p \to \mathbb{G}$. Let $Z \coloneqq f(X_1, X_2)$, where $f : (\{0,1\}^\lambda)^{2^m} \times \mathbb{G}^p \to \{0,1\}^S$ is an arbitrary function. For any $\gamma > 0$, there exists a family $\{(Y_1, Y_2)^{Z, \mathcal{Y}}\}$, indexed by $Z \in \{0,1\}^S$ and size-p subsets $\mathcal{Y}$ of $\mathbb{G}$, of convex combinations of (P_1, P_2)-Bit-Fixing (H, τ)-sources satisfying $P_1\lambda + P_2 \log(p/e) \leq W$ such that for any distinguisher $\mathcal{D}$ taking an S-bit input and querying at most $T_1 < P_1$ coordinates and $T_2 < P_2$ coordinates of each oracle,*
$$\left|\Pr\left[\mathcal{D}^{X_1,X_2}(f(X_1,X_2)) = 1\right] - \Pr\left[\mathcal{D}^{(Y_1,Y_2)^{f(X_1,X_2),\mathrm{im}(X_2)}}(f(X_1,X_2)) = 1\right]\right| \leq \frac{(S + \log\gamma^{-1})(T_1\lambda + T_2\log p)}{W} + \gamma,$$
and
$$\Pr\left[\mathcal{D}^{X_1,X_2}(f(X_1,X_2)) = 1\right] \leq 2^{(S+\log\gamma^{-1})(T_1\lambda+T_2\log p)/W} \cdot \Pr\left[\mathcal{D}^{(Y_1,Y_2)^{f(X_1,X_2),\mathrm{im}(X_2)}}(f(X_1,X_2)) = 1\right] + 2\gamma,$$
where e is the Euler's number.

Proof. Fix an arbitrary $z \in \{0,1\}^S$ and let (X_1^z, X_2^z) be the distribution of (X_1, X_2) conditioned on $f(X_1, X_2) = z$. Let $S_z = 2^m\lambda + \log p! - H_\infty(X_1, X_2)$ be the min-entropy deficiency of (X_1^z, X_2^z). Let $\gamma > 0$ be arbitrary.

Claim 1. *For every $\delta > 0$, any leaky source (X_1^z, X_2^z) is γ-close to a source (Y_1^z, Y_2^z) which is a convex combination of sources such that each source in the convex combination is $(P_1', P_2', 1-\delta)$-dense sources for some P_1' and P_2' which satisfy*
$$P_1'\lambda + P_2'\log(p/e) \leq \frac{S_z + \log\gamma^{-1}}{\delta}.$$

Proof of Claim 1. Without loss of generality, we can assume that (X_1^z, X_2^z) is not $(1-\delta)$-dense since if it is $(1-\delta)$-dense then we can trivially find such $(P_1', P_2', 1-\delta)$-dense sources. Let $(Y_1^z, Y_2^z) \coloneqq (X_1^z, X_2^z)$ and $I_1 \subseteq \{0,1\}^m$ and $I_2 \subseteq \mathbb{Z}_p$ be the largest subset for which there exists a violation, i.e., there exists $y_1^{I_1} \in \left(\{0,1\}^\lambda\right)^{|I_1|}$ and $y_2^{I_2} \in \mathbb{G}^{|I_2|}$ such that

$$\Pr\left[(Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2}\right] > 2^{-(1-\delta)\left[|I_1|\lambda + \log p^{\underline{|I_2|}}\right]}.$$

Let $(\widetilde{Y}_1^z, \widetilde{Y}_2^z)$ be the distribution of (Y_1^z, Y_2^z) conditioned on the event $(Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2}$.

(1) Now we claim that $(\widetilde{Y}_1^z, \widetilde{Y}_2^z)$ is $(P_1', P_2', 1-\delta)$-dense with $P_1' = |I_1|$ and $P_2' = |I_2|$. Suppose that $(\widetilde{Y}_1^z, \widetilde{Y}_2^z)$ is not $(P_1', P_2', 1-\delta)$-dense. Then there exists a pair of non-empty sets $J_1 \subseteq \overline{I_1}$ and $J_2 \subseteq \overline{I_2}$ and $(y_1^{J_1}, y_2^{J_2})$ such that

$$\begin{aligned}
&\Pr\left[(\widetilde{Y}_1^z)_{J_1} = y_1^{J_1} \wedge (\widetilde{Y}_2^z)_{J_2} = y_2^{J_2}\right] \\
&= \Pr\left[(Y_1^z)_{J_1} = y_1^{J_1} \wedge (Y_2^z)_{J_2} = y_2^{J_2} \;\middle|\; (Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2}\right] \\
&> 2^{-(1-\delta)\left[|J_1|\lambda + \log p^{\underline{|J_2|}}\right]}.
\end{aligned}$$

The set $I_1 \cup J_1$ and $I_2 \cup J_2$ now form subsets for which

$$\begin{aligned}
&\Pr\left[(Y_1^z)_{I_1 \cup J_1} = y_1^{I_1 \cup J_1} \wedge (Y_2^z)_{I_2 \cup J_2} = y_2^{I_2 \cup J_2}\right] \\
&= \Pr\left[(Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_1^z)_{J_1} = y_1^{J_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2} \wedge (Y_2^z)_{J_2} = y_2^{J_2}\right] \\
&= \Pr\left[(Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2}\right] \\
&\quad \cdot \Pr\left[(Y_1^z)_{J_1} = y_1^{J_1} \wedge (Y_2^z)_{J_2} = y_2^{J_2} \;\middle|\; (Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2}\right] \\
&> 2^{-(1-\delta)\left[|I_1|\lambda + \log p^{\underline{|I_2|}}\right]} \cdot 2^{-(1-\delta)\left[|J_1|\lambda + \log p^{\underline{|J_2|}}\right]} \\
&= 2^{-(1-\delta)\left[|I_1 \cup J_1|\lambda + \log p^{\underline{|I_2 \cup J_2|}}\right]},
\end{aligned}$$

since I_i's and J_i's are disjoint for $i = 1, 2$. However, this contradicts the maximality of I_1 and I_2.

(2) Next, we claim that $|I_1|\lambda + |I_2|\log(p/e) \le S_z/\delta$. Let $\mathcal{B}_{y_2^{I_2}}$ be the set of vectors $y_2^{\overline{I_2}}$ such that $y_2^{I_2}$ and $y_2^{\overline{I_2}}$ are a valid injection. Since $H_\infty(Y_1^z, Y_2^z) \ge 2^m\lambda + \log p! - S_z$, we observe that for any $y_1^{I_1}$ and $y_2^{I_2}$,

$$\begin{aligned}
&\Pr\left[(Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2}\right] \\
&= \sum_{\substack{y_1^{\overline{I_1}} \in (\{0,1\}^\lambda)^{2^m - |I_1|}, \\ y_2^{\overline{I_2}} \in \mathcal{B}_{y_2^{I_2}}}} \Pr\left[(Y_1^z)_{I_1} = y_1^{I_1} \wedge (Y_2^z)_{I_2} = y_2^{I_2} \wedge (Y_1^z)_{\overline{I_1}} = y_1^{\overline{I_1}} \wedge (Y_2^z)_{\overline{I_2}} = y_2^{\overline{I_2}}\right] \\
&\le 2^{(2^m - |I_1|)\lambda} \cdot 2^{\log(p - |I_2|)!} \cdot 2^{-(2^m\lambda + \log p! - S_z)} \\
&= 2^{-(|I_1|\lambda + \log p^{\underline{|I_2|}} - S_z)},
\end{aligned}$$

and, hence,

$$H_\infty((Y_1^z)_{I_1}, (Y_2^z)_{I_2}) \ge |I_1|\lambda + \log p^{\underline{|I_2|}} - S_z. \tag{2}$$

On the other hand, because $((Y_1^z)_{I_1}, (Y_2^z)_{I_2})$ is not $(1-\delta)$-dense, we have

$$H_\infty((Y_1^z)_{I_1}, (Y_2^z)_{I_2}) < (1-\delta)\left[|I_1|\lambda + \log p^{\underline{|I_2|}}\right]. \tag{3}$$

Combining equation (2) and (3) together, we have

$$\begin{aligned}
S_z &\ge \delta\left[|I_1|\lambda + \log p^{\underline{|I_2|}}\right] \\
&\ge \delta\left[|I_1|\lambda + |I_2|\log(p/e)\right],
\end{aligned}$$

where the last inequality comes from [11, Proposition 38][10]. Divided by $\delta > 0$, we get the result that $|I_1|\lambda + |I_2|\log(p/e) \le S_z/\delta$.

Hence, $(\widetilde{Y}_1^z, \widetilde{Y}_2^z)$ is a $(P_1', P_2', 1-\delta)$-dense source such that $P_1'\lambda + P_2'\log(p/e) \le S_z/\delta$. Set (Y_1^z, Y_2^z) now to be (Y_1^z, Y_2^z) conditioned on $(Y_1^z)_{I_1} \ne y_1^{I_1}$ and $(Y_2^z)_{I_2} \ne y_2^{I_2}$ and recursively decompose (Y_1^z, Y_2^z) as long as

$$\Pr\left[X_1^z \in \mathrm{supp}(Y_1^z) \wedge X_2^z \in \mathrm{supp}(Y_2^z)\right] > \gamma.$$

Observe that $H_\infty(Y_1^z, Y_2^z) \ge 2^m\lambda + \log p! - (S_z + \log\gamma^{-1})$ at any point in this decomposition process since

$$\begin{aligned}
\Pr\left[Y_1^z = y_1 \wedge Y_2^z = y_2\right] &= \Pr\left[X_1^z = y_1 \wedge X_2^z = y_2 | X_1^z \in \mathrm{supp}(Y_1^z) \wedge X_2^z \in \mathrm{supp}(Y_2^z)\right] \\
&\le \frac{\Pr[X_1^z = y_1 \wedge X_2^z = y_2]}{\Pr[\mathrm{supp}(Y_1^z) \wedge X_2^z \in \mathrm{supp}(Y_2^z)]} \\
&\le \frac{2^{-(2^m\lambda + \log p! - S_z)}}{\gamma} \\
&= 2^{-(2^m\lambda + \log p! - (S_z + \log\gamma^{-1}))}.
\end{aligned}$$

Note that $|\mathrm{supp}(Y_1^z)|$ and $|\mathrm{supp}(Y_2^z)|$ decreases in every step, and since $\mathrm{supp}(X_1^z)$ and $\mathrm{supp}(X_2^z)$ are finite, after finitely many steps, this process ends with $(Y_{1,\mathrm{final}}^z, Y_{2,\mathrm{final}}^z)$ with

$$\Pr\left[X_1^z \in \mathrm{supp}(Y_{1,\mathrm{final}}^z) \wedge X_{2,\mathrm{final}}^z \in \mathrm{supp}(Y_2^z)\right] > \gamma.$$

Hence, (X_1^z, X_2^z) is a convex combination of finitely many $(P_1', P_2', 1-\delta)$-dense sources and $(Y_{1,\mathrm{final}}^z, Y_{2,\mathrm{final}}^z)$, where P_1' and P_2' satisfy the inequality

$$P_1'\lambda + P_2'\log(p/e) \le (S_z + \log\gamma^{-1})/\delta.$$

This implies that (X_1^z, X_2^z) is γ-close to a convex combination of $(P_1', P_2', 1-\delta)$-dense sources where $P_1'\lambda + P_2'\log(p/e) \le (S_z + \log\gamma^{-1})/\delta$. □

Let $(\widetilde{X}_1^z, \widetilde{X}_2^z)$ be the convex combination of $(P_1', P_2', 1-\delta)$-dense sources that is γ-close to (X_1^z, X_2^z) for a $\delta = \delta_z$ to be determined later. For every $(P_1', P_2', 1-\delta)$ source $(\widetilde{X}_1, \widetilde{X}_2)$ in said convex combination, let $(\widetilde{Y}_1, \widetilde{Y}_2)$ be the corresponding (P_1', P_2')-Bit-Fixing source, i.e., $(\widetilde{X}_1, \widetilde{X}_2)$ and $(\widetilde{Y}_1, \widetilde{Y}_2)$ are fixed on the same coordinates to the same values. The following claim bounds the distinguishing advantage between $(\widetilde{X}_1, \widetilde{X}_2)$ and $(\widetilde{Y}_1, \widetilde{Y}_2)$ for any (T_1, T_2)-query distinguisher.

Claim 2. *For any $(P_1', P_2', 1-\delta)$-dense source $(\widetilde{X}_1, \widetilde{X}_2)$ and its corresponding (P_1', P_2')-Bit-Fixing source $(\widetilde{Y}_1, \widetilde{Y}_2)$, it holds that for any (adaptive) distinguisher $\mathcal{D}$ that queries at most T_1 coordinates of the random oracle H and makes at most T_2 forward or inverse queries to the injection map τ,*

$$\left|\Pr\left[\mathcal{D}^{\widetilde{X}_1, \widetilde{X}_2} = 1\right] - \Pr\left[\mathcal{D}^{\widetilde{Y}_1, \widetilde{Y}_2} = 1\right]\right| \le \delta(T_1\lambda + T_2\log p),$$

and

$$\Pr\left[\mathcal{D}^{\widetilde{X}_1, \widetilde{X}_2} = 1\right] \le 2^{\delta(T_1\lambda + T_2\log p)} \cdot \Pr\left[\mathcal{D}^{\widetilde{Y}_1, \widetilde{Y}_2} = 1\right].$$

[10] [11, Proposition 38] says that $N^{\underline{j}} \ge (N/e)^j$, where $a^{\underline{b}} \coloneqq a!/(a-b)!$.

Proof of Claim 2. Without loss of generality, assume that $\mathcal{D}$ is deterministic and does not query any of the fixed positions, make the same query twice, or make an inverse query after making the corresponding forward query or vice-versa. Let $T_{\widetilde{X}_1,\widetilde{X}_2}$ and $T_{\widetilde{Y}_1,\widetilde{Y}_2}$ be the random variables corresponding to the transcripts containing the query/answer pairs resulting from $\mathcal{D}$'s interaction with $(\widetilde{X}_1,\widetilde{X}_2)$ and $(\widetilde{Y}_1,\widetilde{Y}_2)$, respectively.

For a fixed transcript t, denote by $\mathrm{p}_{\widetilde{X}_1,\widetilde{X}_2}(t)$ and $\mathrm{p}_{\widetilde{Y}_1,\widetilde{Y}_2}(t)$ the probabilities that $(\widetilde{X}_1,\widetilde{X}_2)$ and $(\widetilde{Y}_1,\widetilde{Y}_2)$, respectively, produce the answers in t if the queries in t are asked. Since $\mathcal{D}$ is deterministic, $\Pr\left[T_{\widetilde{X}_1,\widetilde{X}_2}=t\right]\in\left\{0,\mathrm{p}_{\widetilde{X}_1,\widetilde{X}_2}(t)\right\}$, and similarly, $\Pr\left[T_{\widetilde{Y}_1,\widetilde{Y}_2}=t\right]\in\left\{0,\mathrm{p}_{\widetilde{Y}_1,\widetilde{Y}_2}(t)\right\}$. Denote by $\mathcal{T}_{X_1,X_2}$ the set of all transcripts t for which $\Pr\left[T_{\widetilde{X}_1,\widetilde{X}_2}=t\right]>0$. For such t, $\Pr\left[T_{\widetilde{X}_1,\widetilde{X}_2}=t\right]=\mathrm{p}_{\widetilde{X}_1,\widetilde{X}_2}(t)$ and also $\Pr\left[T_{\widetilde{Y}_1,\widetilde{Y}_2}=t\right]=\mathrm{p}_{\widetilde{Y}_1,\widetilde{Y}_2}(t)$. Observe that for every transcript t,

$$\mathrm{p}_{\widetilde{X}_1,\widetilde{X}_2}(t)\le 2^{-(1-\delta)\left[T_1\lambda+\log(p-P_2')^{\underline{T_2}}\right]}\text{ and }\mathrm{p}_{\widetilde{Y}_1,\widetilde{Y}_2}(t)\le 2^{-\left[T_1\lambda+\log(p-P_2')^{\underline{T_2}}\right]},\tag{4}$$

as $(\widetilde{X}_1,\widetilde{X}_2)$ is $(P_1',P_2',1-\delta)$-dense and $(\widetilde{Y}_1,\widetilde{Y}_2)$ is (P_1',P_2')-fixed.

Towards proving the first part of the claim, observe that $\mathcal{D}$'s output can be computed from the transcript (including whether a query was a forward for an inverse query) by just running $\mathcal{D}$ and providing the answers to its queries from the transcript. Hence,

$$\begin{aligned}\left|\Pr\left[\mathcal{D}^{\widetilde{X}_1,\widetilde{X}_2}=1\right]-\Pr\left[\mathcal{D}^{\widetilde{Y}_1,\widetilde{Y}_2}=1\right]\right|&\le \mathrm{SD}\left(T_{\widetilde{X}_1,\widetilde{X}_2},T_{\widetilde{Y}_1,\widetilde{Y}_2}\right)\\ &=\sum_t\max\left\{0,\Pr\left[T_{\widetilde{X}_1,\widetilde{X}_2}=t\right]-\Pr\left[T_{\widetilde{Y}_1,\widetilde{Y}_2}=t\right]\right\}\\ &=\sum_{t\in\mathcal{T}_{X_1,X_2}}\max\left\{0,\mathrm{p}_{\widetilde{X}_1,\widetilde{X}_2}(t)-\mathrm{p}_{\widetilde{Y}_1,\widetilde{Y}_2}(t)\right\}\\ &=\sum_{t\in\mathcal{T}_{X_1,X_2}}\mathrm{p}_{\widetilde{X}_1,\widetilde{X}_2}(t)\cdot\max\left\{0,1-\frac{\mathrm{p}_{\widetilde{Y}_1,\widetilde{Y}_2}(t)}{\mathrm{p}_{\widetilde{X}_1,\widetilde{X}_2}(t)}\right\}\\ &\le 1-2^{-\delta\left[T_1\lambda+\log(p-P_2')^{\underline{T_2}}\right]}\\ &\le 1-2^{-\delta(T_1\lambda+T_2\log p)}\le\delta(T_1\lambda+T_2\log p),\end{aligned}$$

where the first sum is over all possible transcripts and where the last inequality uses $2^{-x}\ge 1-x$ for $x\ge 0$ and $a^{\underline{b}}\le a^b$ for $a,b\in\mathbb{N}$.

As for the second part of the claim, observe that due to the equation (4) and the support of $T_{\widetilde{X}_1,\widetilde{X}_2}$ being a subset of $T_{\widetilde{Y}_1,\widetilde{Y}_2}$,

$$\begin{aligned}\Pr\left[T_{\widetilde{X}_1,\widetilde{X}_2}=t\right]&\le 2^{\delta\left[T_1\lambda+\log(p-P_2')^{\underline{T_2}}\right]}\cdot\Pr\left[T_{\widetilde{Y}_1,\widetilde{Y}_2}=t\right]\\ &\le 2^{\delta(T_1\lambda+T_2\log p)}\cdot\Pr\left[T_{\widetilde{Y}_1,\widetilde{Y}_2}=t\right],\end{aligned}$$

for any transcript t. Let $\mathcal{T}_{\mathcal{D}}$ be the set of transcripts where $\mathcal{D}$ outputs 1. Then we have

$$\begin{aligned}\Pr\left[\mathcal{D}^{\widetilde{X}_1,\widetilde{X}_2}=1\right]&=\sum_{t\in\mathcal{T}_{\mathcal{D}}}\Pr\left[T_{\widetilde{X}_1,\widetilde{X}_2}=t\right]\\ &\le 2^{\delta(T_1\lambda+T_2\log p)}\cdot\sum_{t\in\mathcal{T}_{\mathcal{D}}}\Pr\left[T_{\widetilde{Y}_1,\widetilde{Y}_2}=t\right]\\ &=2^{\delta(T_1\lambda+T_2\log p)}\cdot\Pr\left[\mathcal{D}^{\widetilde{Y}_1,\widetilde{Y}_2}=1\right],\end{aligned}$$

which concludes the proof. □

Let $(\widetilde{Y}_1^z,\widetilde{Y}_2^z)$ be obtained by replacing $(\widetilde{X}_1,\widetilde{X}_2)$ by $(\widetilde{Y}_1,\widetilde{Y}_2)$ in $(\widetilde{X}_1^z,\widetilde{X}_2^z)$. Setting

$$\delta_z=\frac{S_z+\log\gamma^{-1}}{P_1k+P_2\log(p/e)},$$

Claim 1 and Claim 2 imply

$$\begin{aligned}&\left|\Pr\left[\mathcal{D}^{X_1^z,X_2^z}(z)=1\right]-\Pr\left[\mathcal{D}^{\widetilde{Y}_1^z,\widetilde{Y}_2^z}(z)=1\right]\right|\\ &\le\frac{(S_z+\log\gamma^{-1})(T_1\lambda+T_2\log p)}{(P_1\lambda+P_2\log(p/e))}+\gamma,\end{aligned}\tag{5}$$

as well as

$$\begin{aligned}&\Pr\left[\mathcal{D}^{X_1^z,X_2^z}(z)=1\right]\\ &\le 2^{\frac{(S_z+\log\gamma^{-1})(T_1\lambda+T_2\log p)}{(P_1\lambda+P_2\log(p/e))}}\cdot\Pr\left[\mathcal{D}^{\widetilde{Y}_1^z,\widetilde{Y}_2^z}(z)=1\right]+\gamma.\end{aligned}\tag{6}$$

Moreover, note that for the above choice of δ_z, we have $P_1'=P_1$ and $P_2'=P_2$, i.e., the sources $(\widetilde{Y}_1^z,\widetilde{Y}_2^z)$ are (P_1,P_2)-fixed, as desired.

Claim 3. *$\mathbb{E}_z[S_z]\le S$ and $\Pr[S_{f(X_1,X_2)}>S+\log\gamma^{-1}]\le\gamma$.*

Proof of Claim 3. Observe that $H_\infty(X_1^z,X_2^z)=H_\infty((X_1,X_2)|Z=z)=H((X_1,X_2)|Z=z)$ since, conditioned on $Z=z$, (X_1,X_2) is distributed uniformly over all values (x_1,x_2) with $f(x_1,x_2)=z$. Hence,

$$\begin{aligned}\mathbb{E}_z[S_z]&=2^m\lambda+\log p!-\mathbb{E}_z\left[H_\infty((X_1,X_2)|Z=z)\right]\\ &=2^m\lambda+\log p!-\mathbb{E}_z\left[H((X_1,X_2)|Z=z)\right]\\ &=2^m\lambda+\log p!-H((X_1,X_2)|Z=z)\le S.\end{aligned}$$

Again, due to the uniformity of (X_1,X_2), $\Pr[f(X_1,X_2)=z]=2^{-S_z}$. Hence,

$$\begin{aligned}\Pr[S_{f(X_1,X_2)}>S+\log\gamma^{-1}]&=\sum_{z\in\{0,1\}^S:S_z>S+\log\gamma^{-1}}\Pr[f(X_1,X_2)=z]\\ &\le 2^S\cdot 2^{-(S+\log\gamma^{-1})}\le\gamma.\end{aligned}$$ □

Now the first part of the lemma follows (using $(Y_1^z,Y_2^z):=(\widetilde{Y}_1^z,\widetilde{Y}_2^z)$) by taking expectations over z of the equation (5) and applying the first part of Claim 3. The second part of the lemma is proved as follows:

$$\begin{aligned}&\Pr\left[\mathcal{D}^{X_1,X_2}(f(X_1,X_2))=1\right]\\ &\le\Pr\left[\mathcal{D}^{X_1,X_2}(f(X_1,X_2))=1,S_{f(X_1,X_2)}\le S+\log\gamma^{-1}\right]\\ &\qquad+\Pr\left[S_{f(X_1,X_2)}>S+\log\gamma^{-1}\right]\\ &\le\Big(2^{(S+\log\gamma^{-1})(T_1\lambda+T_2\log p)/W}\cdot\Pr\Big[\mathcal{D}^{X_1,X_2}(f(X_1,X_2))=1,\\ &\qquad S_{f(X_1,X_2)}\le S+\log\gamma^{-1}\Big]+\gamma\Big)+\gamma\\ &\le 2^{(S+\log\gamma^{-1})(T_1\lambda+T_2\log p)/W}\cdot\Pr\Big[\mathcal{D}^{X_1,X_2}(f(X_1,X_2))=1,\\ &\qquad S_{f(X_1,X_2)}\le S+\log\gamma^{-1}\Big]+2\gamma,\end{aligned}$$

where the second inequality follows by taking expectations over z of the equation (6) (together with the condition $S_z\le S+\log\gamma^{-1}$) and the second part of Claim 3. □

A.2 From the BF-ROM+GGM to the AI-ROM+GGM

Capturing the Models. An oracle pair (O_1, O_2) has two interfaces $(O_1.\mathsf{pre}, O_2.\mathsf{pre})$ and $(O_1.\mathsf{on}, O_2.\mathsf{on})$, where $(O_1.\mathsf{pre}, O_2.\mathsf{pre})$ is accessible only once before any calls to $(O_1.\mathsf{on}, O_2.\mathsf{on})$ are made. We consider the oracles in this work as follows:

- **Random Oracle + Generic Group Oracle** $\mathsf{RO{+}GG}(m, \lambda, p, \ell)$: Samples a random oracle function table $\mathsf{H} \leftarrow \mathcal{H}_{m,\lambda}$, where $\mathcal{H}_{m,\lambda}$ is the set of all functions from $\{0,1\}^m \to \{0,1\}^\lambda$, and samples a random injection $\tau \leftarrow \mathcal{I}_{p,\ell}$, where $\mathcal{I}_{p,\ell}$ is the set of all injections from $\mathbb{Z}_p \to \mathbb{G}$ where $\mathbb{G}$ is the set of bitstrings of length $\ell \geq \log p$; offers no functionality at $(O_1.\mathsf{pre}, O_2.\mathsf{pre})$; answers queries $x \in \{0,1\}^m$ to H at $O_1.\mathsf{on}$ by the corresponding value $\mathsf{H}(x) \in \{0,1\}^\lambda$; answers *forward* queries $x' \in \mathbb{Z}_p$ to τ at $O_2.\mathsf{on}$ by the corresponding value $\tau(x') \in \mathbb{G}$; answers *group-operation* queries (s, s') at $O_2.\mathsf{on}$ as follows: if $s = \tau(x)$ and $s' = \tau(x')$ for some x, x', the oracle replies by $\tau(x + x')$ and by $\perp$ otherwise; answers *inverse* queries s at $O_2.\mathsf{on}$ by returning $\tau^{-1}(s)$ if s is in the range of τ and $\perp$ otherwise.
- **Auxiliary-Input Random Oracle + Generic Group Oracle** $\mathsf{AI\text{-}RO{+}GG}(m, \lambda, p, \ell)$: Samples a random oracle function table $\mathsf{H} \leftarrow \mathcal{H}_{m,\lambda}$ and samples a random injection $\tau \leftarrow \mathcal{I}_{p,\ell}$ as explained in $\mathsf{RO{+}GG}(m, \lambda, p, \ell)$; outputs all of H at $O_1.\mathsf{pre}$ and all of τ at $O_2.\mathsf{pre}$; $(O_1.\mathsf{on}, O_2.\mathsf{on})$ behaves the same as $\mathsf{RO{+}GG}(m, \lambda, p, \ell)$. When the parameters (m, λ, p, ℓ) are clear in context, we sometimes abuse the notation and simply say $\mathsf{AI\text{-}RO{+}GG}$.
- **Bit-Fixing Random Oracle + Generic Group Oracle** $\mathsf{BF\text{-}RO{+}GG}(P_1, P_2, m, \lambda, p, \ell)$: Samples a random oracle function table $\mathsf{H} \leftarrow \mathcal{H}_{m,\lambda}$ at $O_1.\mathsf{pre}$ and samples a random size-p subset $\mathcal{Y}$ of $\mathbb{G}$ and outputs $\mathcal{Y}$ at $O_2.\mathsf{pre}$; takes a list at $(O_1.\mathsf{pre}, O_2.\mathsf{pre})$ of at most P_1 query/answer pairs that override H in the corresponding positions and at most P_2 query/answer pairs without collisions and all answers in $\mathcal{Y}$; samples a random injection $\tau \leftarrow \mathcal{I}_{p,\ell}$ with range $\mathcal{Y}$ and consistent with said list; $(O_1.\mathsf{on}, O_2.\mathsf{on})$ behaves the same as $\mathsf{RO{+}GG}(m, \lambda, p, \ell)$. When the parameters (m, λ, p, ℓ) are clear in context, we sometimes abuse the notation and simply say $\mathsf{BF\text{-}RO{+}GG}(P_1, P_2)$.

Attackers with Oracle-Dependent Advice. We define the attackers $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$ similar to that of prior work [11, 12], which consist of a preprocessing attacker $\mathcal{A}_{\mathsf{pre}}$ and an online attacker $\mathcal{A}_{\mathsf{on}}$, which carries out the actual attack using the output of $\mathcal{A}_{\mathsf{pre}}$. The difference is that we consider a pair of oracles instead of a single oracle. More precisely, in the presence of an oracle pair (O_1, O_2), $\mathcal{A}_{\mathsf{pre}}$ interacts with $(O_1.\mathsf{pre}, O_2.\mathsf{pre})$ and $\mathcal{A}_{\mathsf{on}}$ with $(O_1.\mathsf{on}, O_2.\mathsf{on})$.

Definition A.3. An *(S, T_1, T_2)-attacker $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$ in the (O_1, O_2)-model* consists of two procedures

- $\mathcal{A}_{\mathsf{pre}}$, which is computationally unbounded, interacts with $(O_1.\mathsf{pre}, O_2.\mathsf{pre})$, and outputs an S-bit string as hint, and
- $\mathcal{A}_{\mathsf{on}}$, which takes an S-bit auxiliary input and makes at most T_1 queries to $O_1.\mathsf{on}$ and T_2 queries to $O_2.\mathsf{on}$.

For an arbitrary oracle O, Coretti et al. [11, 12] defined an *application G* in the O-model by specifying a challenger C that has access to $O.\mathsf{on}$, interacts with the online attacker $\mathcal{A}_{\mathsf{on}}$ of an attacker $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$, and outputs a bit at the end of the interaction. Then they defined the *success* of $\mathcal{A}$ on G in the O-model as follows:

$$\mathrm{Succ}_{G,O}(\mathcal{A}) \coloneqq \Pr\left[\mathcal{A}_{\mathsf{on}}^{O.\mathsf{on}}\left(\mathcal{A}_{\mathsf{pre}}^{O.\mathsf{pre}}\right) \leftrightarrow \mathsf{C}^{O.\mathsf{on}} = 1\right],$$

where $\mathcal{A}_{\mathsf{on}}^{O.\mathsf{on}}\left(\mathcal{A}_{\mathsf{pre}}^{O.\mathsf{pre}}\right) \leftrightarrow \mathsf{C}^{O.\mathsf{on}}$ denotes the bit output by the challenger C after its interaction with $\mathcal{A}$. It can be extended to the oracle pair setting in the same manner when $O = (O_1, O_2)$, by defining $O.\mathsf{pre} \coloneqq (O_1.\mathsf{pre}, O_2.\mathsf{pre})$ and $O.\mathsf{on} \coloneqq (O_1.\mathsf{on}, O_2.\mathsf{on})$.

Definition A.4. For an *indistinguishability* application G in the (O_1, O_2)-model, the *advantage* of an attacker $\mathcal{A}$ is defined as

$$\mathrm{Adv}_{G,(O_1,O_2)}(\mathcal{A}) \coloneqq 2\left|\mathrm{Succ}_{G,(O_1,O_2)}(\mathcal{A}) - \frac{1}{2}\right|.$$

In the case of an *unpredictability* application G, the advantage is defined as $\mathrm{Adv}_{G,(O_1,O_2)}(\mathcal{A}) \coloneqq \mathrm{Succ}_{G,(O_1,O_2)}(\mathcal{A})$. An application G is said to be *$(S, T_1, T_2, \varepsilon)$-secure* in the (O_1, O_2)-model if for every (S, T_1, T_2)-attacker $\mathcal{A}$,

$$\mathrm{Adv}_{G,(O_1,O_2)}(\mathcal{A}) \leq \varepsilon.$$

Additive Error for Arbitrary Applications in the ROM+GGM.. Using Lemma A.2, one can extend [12, Theorem 5] and [11, Theorem 1] to the translation from the Bit-Fixing model with multiple idealized models (ROM+GGM) to the corresponding Auxiliary-Input model at the cost of an additive term which is similar to those from [12, Theorem 5] and [11, Theorem 1].

Reminder of Theorem 3.1. *For every $\gamma > 0$, if an application G is $(S, T_1, T_2, \varepsilon')$-secure in the $\mathsf{BF\text{-}RO{+}GG}(P_1, P_2, m, \lambda, p, \ell)$-model for any $P_1, P_2 \in \mathbb{N}$ such that $P_1\lambda + P_2 \log(p/e) \leq \eta$, then it is $(S, T_1, T_2, \varepsilon)$-secure in the $\mathsf{AI\text{-}RO{+}GG}(m, \lambda, p, \ell)$-model, for*

$$\varepsilon \leq \varepsilon' + \frac{2(S + \log \gamma^{-1})\left((T_1)_G^{\mathrm{comb}}\lambda + 3(T_2)_G^{\mathrm{comb}}\log p\right)}{\eta} + 2\gamma,$$

where e is the Euler constant, and $(T_1)_G^{\mathrm{comb}}$ and $(T_2)_G^{\mathrm{comb}}$ are the combined query complexity corresponding to G that corresponds to the random oracle and the generic group oracle, respectively.

Proof. Fix P_1, P_2 as well as γ. Let G be an arbitrary application in the $(\mathsf{BF\text{-}RO}, \mathsf{BF\text{-}GG})$-model (or in the $(\mathsf{AI\text{-}RO}, \mathsf{AI\text{-}GG})$-model) and C be the corresponding challenger. Moreover, fix an (S, T_1, T_2)-attacker $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$, and let $\{(Y_1, Y_2)^{z,\mathcal{Y}}\}$, indexed by $z \in \{0,1\}^S$ and size-p subsets $\mathcal{Y} \subseteq \mathbb{G}$, be the family of distributions guaranteed to exist by Lemma A.2. Consider the following (S, T_1, T_2)-attacker $\mathcal{A}' = (\mathcal{A}'_{\mathsf{pre}}, \mathcal{A}'_{\mathsf{on}})$ (expecting to interact with $\mathsf{BF\text{-}RO{+}GG}$):

- $\mathcal{A}'_{\mathsf{pre}}$ obtains the set $\mathcal{Y}$ from $\mathsf{BF\text{-}GG.pre}$ and internally simulates $\mathcal{A}_{\mathsf{pre}}$ on a uniformly random input $X_1 \in \{0,1\}^m$ and a uniformly random injection $X_2 \in \mathbb{Z}_p$ with range $\mathcal{Y}$ to obtain $z \leftarrow \mathcal{A}_{\mathsf{pre}}^{\mathsf{AI\text{-}RO{+}GG.pre}}$. Then, it samples one of the (P_1, P_2)-bit-fixing sources (Y_1', Y_2') making up $(Y_1, Y_2)^{z,\mathcal{Y}}$ and presets $\mathsf{BF\text{-}RO{+}GG}$ to match (Y_1', Y_2') on the at most P_1 (resp. P_2) points where Y_1' (resp. Y_2') is fixed. The output of $\mathcal{A}'_{\mathsf{pre}}$ is z.
- $\mathcal{A}'_{\mathsf{on}}$ works exactly the same as $\mathcal{A}_{\mathsf{on}}$.

Let $\mathcal{D}$ be a distinguisher — making forward, group-operation, and inverse queries to a (H, τ)-source — that internally runs the combination of $\mathcal{A}_{\text{on}} = \mathcal{A}'_{\text{on}}$ and C. It answers their queries as follows:

(1) It passes forward queries to and back from its own oracle.
(2) It answers group-operation queries (s, s') (to the generic group oracle) by making two backward queries to its own oracle for s and s', obtaining i and j, respectively, making a forward query $i + j$, and passing the answer to $\mathcal{A}_{\text{on}}$ or C (unless one of the answers to the backward queries was $\perp$, in which case $\perp$ is returned).

Note that $\mathcal{D}^{X_1,X_2}(f(X_1, X_2))$ is identical to

$$\mathcal{A}_{\text{on}}^{\text{AI-RO+GG.on}}(\mathcal{A}_{\text{pre}}^{\text{AI-RO+GG.pre}}) \leftrightarrow \mathsf{C}^{\text{AI-RO+GG.on}},$$

and $\mathcal{D}^{(Y_1,Y_2)^{f(X_1,X_2),\text{im}(X_2)}}(f(X_1, X_2))$ is identical to

$$\mathcal{A}'^{\text{BF-RO+GG.on}}_{\text{on}}(\mathcal{A}'^{\text{BF-RO+GG.pre}}_{\text{pre}}) \leftrightarrow \mathsf{C}^{\text{BF-RO+GG.on}}.$$

Furthermore, $\mathcal{D}$ is a distinguisher taking an S-bit input and making at most $(T_1)_G^{\text{comb}}$ to its random oracle and $3(T_2)_G^{\text{comb}}$ to its generic group oracle. Therefore, by Lemma A.2,

$$\begin{aligned}\text{Succ}_{G,\text{AI-RO+GG}}(\mathcal{A}) &\le \text{Succ}_{G,\text{BF-RO+GG}}(\mathcal{A}') \\ &\quad + \frac{(S + \log\gamma^{-1})\left((T_1)_G^{\text{comb}}\lambda + 3(T_2)_G^{\text{comb}}\log p\right)}{\eta} + \gamma,\end{aligned}$$

for any P_1, P_2 satisfying $P_1\lambda + P_2 \log(p/e)$. Since there is only an additive term between the two success probabilities, the above inequality implies

$$\begin{aligned}\text{Adv}_{G,\text{AI-RO+GG}}(\mathcal{A}) &\le \text{Adv}_{G,\text{BF-RO+GG}}(\mathcal{A}') \\ &\quad + \frac{2(S + \log\gamma^{-1})\left((T_1)_G^{\text{comb}}\lambda + 3(T_2)_G^{\text{comb}}\log p\right)}{\eta} + 2\gamma,\end{aligned}$$

for both indistinguishability and unpredictability applications. Note that the extra factor of 2 is technically only necessary for indistinguishability applications [12]. □

Reminder of Theorem 3.2. *For every $\gamma > 0$, if an* unpredictability *application G is $(S, T_1, T_2, \varepsilon')$-secure in the* BF-RO+GG$(P_1, P_2, m, \lambda, p, \ell)$*-model for any $P_1, P_2 \in \mathbb{N}$ satisfying*

$$(S + \log\gamma^{-1})\left((T_1)_G^{\text{comb}}\lambda + 3(T_2)_G^{\text{comb}}\log p\right) \le P_1\lambda + P_2\log(p/e),$$

then it is $(S, T_1, T_2, \varepsilon)$-secure in the AI-RO+GG$(m, \lambda, p, \ell)$*-model, for*

$$\varepsilon \le 2\varepsilon' + 2\gamma,$$

where e is the Euler's number.

Proof. Using the same attacker $\mathcal{A}'$ as in the proof of Theorem 3.1 and applying the second part of Lemma A.2, for any $P_1, P_2 \in \mathbb{N}$ such that $(S + \log\gamma^{-1})\left((T_1)_G^{\text{comb}}\lambda + 3(T_2)_G^{\text{comb}}\log p\right) \le P_1\lambda + P_2\log(p/e) \le \eta$, we have

$$\begin{aligned}&\text{Succ}_{G,\text{AI-RO+GG}}(\mathcal{A}) \\ &\le 2^{\frac{(S+\log\gamma^{-1})\left((T_1)_G^{\text{comb}}\lambda+3(T_2)_G^{\text{comb}}\log p\right)}{\eta}} \cdot \text{Succ}_{G,\text{BF-RO+GG}}(\mathcal{A}') + 2\gamma \\ &\le 2 \cdot \text{Succ}_{G,\text{BF-RO+GG}}(\mathcal{A}') + 2\gamma,\end{aligned}$$

which translates into

$$\text{Adv}_{G,\text{AI-RO+GG}}(\mathcal{A}) \le 2 \cdot \text{Adv}_{G,\text{BF-RO+GG}}(\mathcal{A}') + 2\gamma$$

for unpredictability applications. □

B Bit-Fixing/Auxiliary-Input Multiple Idealized Models

In Section 3, we studied the Bit-Fixing to Auxiliary-Input transition with two idealized models at once — the Random Oracle Model plus the Generic Group Model. We can further extend this approach and consider multiple idealized models, including the Ideal Cipher Model and the Random Permutation Model as well.

B.1 Replacing Auxiliary Information by Bit-Fixing

Definition B.1. A (n_1, n_2, n_3)*-source* is a tuple of vector random variables $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where

- $\mathbf{X} = (X_1, \ldots, X_{n_1})$ is an n_1-dimensional vector where for each $i \in [n_1]$, an element X_i is a random variable that corresponds to the function table of a random oracle $\mathsf{H}_i : [M_i] \to [N_i]$ with range $[N_i]^{M_i}$,
- $\mathbf{Y} = (Y_1, \ldots, Y_{n_2})$ is an n_2-dimensional vector where for each $i \in [n_2]$, an element Y_i is a random variable that corresponds to the function table of an injection $\tau_i : \mathbb{Z}_{p_i} \to \mathbb{G}_i$, where $\mathbb{G}_i$ is the set of bitstrings of length ℓ_i (with $\ell_i \ge \log p_i$), and
- $\mathbf{Z} = (Z_1, \ldots, Z_{n_3})$ is an n_3-dimensional vector where for each $i \in [n_3]$, an element Z_i is a random variable that corresponds to the function table of a cipher $F_i : [K_i] \times [C_i] \to [C_i]$.

Further, a (n_1, n_2, n_3)-source is called:

- $(\mathbf{P}, \mathbf{Q}, \mathbf{R}, 1-\delta)$*-dense* if for vectors $\mathbf{P} = (P_1, \ldots, P_{n_1}) \in \mathbb{Z}_{M_1} \times \cdots \times \mathbb{Z}_{M_{n_1}}$, $\mathbf{Q} = (Q_1, \ldots, Q_{n_2}) \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_{n_1}}$, and $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_{n_3}) \in (\mathbb{Z}_{C_1})^{K_1} \times \cdots \times (\mathbb{Z}_{C_{n_3}})^{K_{n_3}}$ where $\mathbf{R}_k = (R_{k,1}, \ldots, R_{k,K_k})$ for each $k \in [n_3]$, if H_i is fixed on at most P_i coordinates for each $i \in [n_1]$, τ_j is fixed on at most Q_j coordinates for each $j \in [n_2]$, and for each cipher F_k with $k \in [n_3]$, it is fixed on at most $R_{k,s}$ coordinates for each $s \in [K_k]$, and if for all tuples of vectors of subsets $(\mathbf{U}, \mathbf{V}, \mathbf{W})$ of non-fixed coordinates where $\mathbf{U} = (U_1, \ldots, U_{n_1}) \subseteq \mathbb{Z}_{M_1} \times \cdots \times \mathbb{Z}_{M_{n_1}}$, $\mathbf{V} = (V_1, \ldots, V_{n_2}) \subseteq \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_{n_1}}$, and $\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{n_3}) \subseteq (\mathbb{Z}_{C_1})^{K_1} \times \cdots \times (\mathbb{Z}_{C_{n_3}})^{K_{n_3}}$ where $\mathbf{W}_k = (W_{k,1}, \ldots, W_{k,K_k})$ for each $k \in [n_3]$,

$$\begin{aligned}H_\infty(\mathbf{X}_\mathbf{U}, \mathbf{Y}_\mathbf{V}, \mathbf{Z}_\mathbf{W}) \ge (1-\delta)\Bigg[&\sum_{i=1}^{n_1}|U_i|\log N_i + \sum_{j=1}^{n_2}\log(p_j - Q_j)^{\underline{|V_j|}} \\ &+ \sum_{k=1}^{n_3}\sum_{s=1}^{K_k}\log(C_k - R_{k,s})^{\underline{|W_{k,s}|}}\Bigg],\end{aligned}$$

  where $a^{\underline{b}} := a!/(a-b)!$ and $\mathbf{X}_\mathbf{U}$ is $\mathbf{X}$ restricted to the coordinates in $\mathbf{U}$,
- $(1-\delta)$*-dense* if it is $(\mathbf{0}, \mathbf{0}, \mathbf{0}, 1-\delta)$-dense, and
- $(\mathbf{P}, \mathbf{Q}, \mathbf{R})$*-bit-fixing* if it is $(\mathbf{P}, \mathbf{Q}, \mathbf{R}, 1)$-dense, i.e., $(\mathbf{P}, \mathbf{Q}, \mathbf{R}, 1-\delta)$-dense with $\delta = 0$.

In Definition B.1, we remark that $\mathbf{X}$ captures a source with multiple random oracles, $\mathbf{Y}$ captures a source with multiple generic groups, and $\mathbf{Z}$ captures a source with multiple ideal ciphers. In the case of $\mathbf{Z}$, if $K_i = 1$ for some $i \in [n_3]$, it can be considered as a random permutation. Hence, Definition B.1 captures the usage of multiple idealized models at once, including the random oracle model, generic group model, ideal cipher model, and random permutation model.

Lemma B.2. *Let $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ be a tuple of vector random variables which satisfies the following.*

- *$\mathbf{X} = (X_1, \ldots, X_{n_1})$ is an n_1-dimensional vector where X_i is distributed uniformly over $[N_i]^{M_i}$ for each $i \in [n_1]$,*
- *$\mathbf{Y} = (Y_1, \ldots, Y_{n_2})$ is an n_2-dimensional vector where Y_i is a uniformly random injection that takes on as value function tables corresponding to an injection $\tau_i : \mathbb{Z}_{p_i} \to \mathbb{G}_i$ for each $i \in [n_2]$, and*
- *$\mathbf{Z} = (Z_1, \ldots, Z_{n_3})$ is an n_3-dimensional vector where Z_i takes on as value function tables corresponding to a cipher $F_i : [K_i] \times [C_i] \to [C_i]$ for each $i \in [n_3]$.*

Let $W = f(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where

$$f : \Big(\prod_{i=1}^{n_1} [M_i]\Big) \times \Big(\prod_{j=1}^{n_2} \mathbb{Z}_{p_j}\Big) \times \Big(\prod_{k=1}^{n_3} [K_k] \times [C_k]\Big) \to \{0,1\}^S$$

is an arbitrary function. For any $\gamma > 0$ and $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ that were defined as in Definition B.1, there exists a family $\{(\mathbf{X}', \mathbf{Y}', \mathbf{Z}')\}_{W, \mathcal{Y}}$, indexed by $W \in \{0,1\}^S$ and $\mathcal{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_{n_2})$ where $\mathcal{Y}_i$ is a size-p_i subset of $\mathbb{G}_i$ for each $i \in [n_2]$, of convex combinations of $(\mathbf{P}, \mathbf{Q}, \mathbf{R})$-bit-fixing (n_1, n_2, n_3)-sources satisfying

$$\sum_{i=1}^{n_1} P_i \log N_i + \sum_{i=1}^{n_2} Q_i \log(p_i/e) + \sum_{i=1}^{n_3} \sum_{j=1}^{K_i} R_{i,j} \log(C_i/e) \le \eta,$$

such that for any distinguisher $\mathcal{D}$ taking an S-bit input and querying at most $\mathbf{S} = (S_1, \ldots, S_{n_1})$ coordinates of each oracle that corresponds to $\mathbf{X}$ where $S_i < P_i$ for each $i \in [n_1]$, $\mathbf{T} = (T_1, \ldots, T_{n_2})$ coordinates of each oracle that corresponds to $\mathbf{Y}$ where $T_i < Q_i$ for each $i \in [n_2]$, and $\mathbf{L} = (\mathbf{L}_1, \ldots, \mathbf{L}_{n_3})$ coordinates of each oracle that corresponds to $\mathbf{Z}$ where $\mathbf{L}_k = (L_{k,1}, \ldots, L_{k,K_k})$ and $L_{i,j} < R_{i,j}$ for each $j \in [K_i]$ and $i \in [n_3]$, we have (here, e is the Euler's number)

$$\Big|\Pr\Big[\mathcal{D}^{\mathbf{X},\mathbf{Y},\mathbf{Z}}(f(\mathbf{X},\mathbf{Y},\mathbf{Z})) = 1\Big] - \Pr\Big[\mathcal{D}^{(\mathbf{X}',\mathbf{Y}',\mathbf{Z}')_{f(\mathbf{X},\mathbf{Y},\mathbf{Z}),\mathrm{im}(\mathbf{Y})}}(f(\mathbf{X},\mathbf{Y},\mathbf{Z})) = 1\Big]\Big|$$
$$\le \frac{(S + \log\gamma^{-1})\left(\sum_{i=1}^{n_1} S_i \log N_i + \sum_{i=1}^{n_2} T_i \log p_i + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j} \log C_i\right)}{\eta} + \gamma.$$

Proof. Fix an arbitrary $w \in \{0,1\}^S$ and let $(\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$ be the distribution of $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ conditioned on $f(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = w$. Let $S_z = \sum_{i=1}^{n_1} M_i \log N_i + \sum_{i=1}^{n_2} \log p_i! + \sum_{i=1}^{n_3} K_i \log C_i! - H_\infty(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ be the min-entropy deficiency of $(\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$. Let $\gamma > 0$ be arbitrary.

Claim 4. *For every $\delta > 0$, any leaky source $(\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$ is γ-close to a source $(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w)$ which is a convex combination of $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense sources for*

$$\sum_{i=1}^{n_1} P'_i \log N_i + \sum_{i=1}^{n_2} Q'_i \log(p_i/e) + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i} R'_{i,j} \log(C_i/e) \le \frac{S_z + \log\gamma^{-1}}{\delta}.$$

Proof of Claim 4. Without loss of generality, we can assume that $(\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$ is not $(1-\delta)$-dense since if it is $(1-\delta)$-dense then we can trivially find such $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense sources. Let $(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w) \coloneqq (\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$ and $\mathbf{U} = (U_1, \ldots, U_{n_1}) \in \mathbb{Z}_{M_1} \times \cdots \times \mathbb{Z}_{M_{n_1}}$, $\mathbf{V} = (V_1, \ldots, V_{n_2}) \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_{n_1}}$, and $\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{n_3}) \in (\mathbb{Z}_{C_1})^{K_1} \times \cdots \times (\mathbb{Z}_{C_{n_3}})^{K_{n_3}}$ (where $\mathbf{W}_k = (W_{k,1}, \ldots, W_{k,K_k})$ for each $k \in [n_3]$) be the largest subset for which there exists a violation, i.e., there exists $\mathbf{x}'_{\mathbf{U}}, \mathbf{y}'_{\mathbf{V}}, \mathbf{z}'_{\mathbf{W}}$ such that

$$\Pr\Big[(\mathbf{X}'^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}} \wedge (\mathbf{Y}'^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Z}'^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}}\Big]$$
$$> 2^{-(1-\delta)\left[\sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V_j|}} + \sum_{k=1}^{n_3}\sum_{j=1}^{K_k} \log C_k^{\underline{|W_{k,j}|}}\right]}.$$

Let $(\widetilde{\mathbf{X}'}^w, \widetilde{\mathbf{Y}'}^w, \widetilde{\mathbf{Z}'}^w)$ be the distribution of $(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w)$ conditioned on the event $(\mathbf{X}'^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}} \wedge (\mathbf{Y}'^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Z}'^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}}$.

(1) Now we claim that $(\widetilde{\mathbf{X}'}^w, \widetilde{\mathbf{Y}'}^w, \widetilde{\mathbf{Z}'}^w)$ is $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense with $P'_i = |U_i|$ for $i \in [n_1]$, $Q'_j = |V_j|$ for $j \in [n_2]$, and $R'_{k,j} = |W_{k,j}|$ for $k \in [n_3], j \in [K_k]$. Suppose that $(\widetilde{\mathbf{X}'}^w, \widetilde{\mathbf{Y}'}^w, \widetilde{\mathbf{Z}'}^w)$ is not $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense. Then there exists a pair of non-empty of sets $\mathbf{U}' \subseteq \overline{\mathbf{U}}$ (here, $\subseteq$ refers to an pairwise subset notation), $\mathbf{V}' \subseteq \overline{\mathbf{V}}$, and $\mathbf{W}' \subseteq \overline{\mathbf{W}}$ such that

$$\Pr\Big[(\widetilde{\mathbf{X}'}^w)_{\mathbf{U}'} = \mathbf{x}'_{\mathbf{U}'} \wedge (\widetilde{\mathbf{Y}'}^w)_{\mathbf{V}'} = \mathbf{y}'_{\mathbf{V}'} \wedge (\widetilde{\mathbf{Z}'}^w)_{\mathbf{W}'} = \mathbf{z}'_{\mathbf{W}'}\Big]$$
$$= \Pr\Big[(\mathbf{X}'^w)_{\mathbf{U}'} = \mathbf{x}'_{\mathbf{U}'} \wedge (\mathbf{Y}'^w)_{\mathbf{V}'} = \mathbf{y}'_{\mathbf{V}'} \wedge (\mathbf{Z}'^w)_{\mathbf{W}'} = \mathbf{z}'_{\mathbf{W}'} \Big|$$
$$(\mathbf{X}'^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}} \wedge (\mathbf{Y}'^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Z}'^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}}\Big]$$
$$> 2^{-(1-\delta)\left[\sum_{i=1}^{n_1} |U'_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V'_j|}} + \sum_{k=1}^{n_3}\sum_{j=1}^{K_k} \log C_k^{\underline{|W'_{k,j}|}}\right]}.$$

The set $\mathbf{U} \cup \mathbf{U}'$ (also pairwise), $\mathbf{V} \cup \mathbf{V}'$, and $\mathbf{W} \cup \mathbf{W}'$ now form subsets for which

$$\Pr\Big[(\mathbf{X}'^w)_{\mathbf{U}\cup\mathbf{U}'} = \mathbf{x}'_{\mathbf{U}\cup\mathbf{U}'} \wedge (\mathbf{Y}'^w)_{\mathbf{V}\cup\mathbf{V}'} = \mathbf{y}'_{\mathbf{V}\cup\mathbf{V}'} \wedge (\mathbf{Z}'^w)_{\mathbf{W}\cup\mathbf{W}'} = \mathbf{z}'_{\mathbf{W}\cup\mathbf{W}'}\Big]$$
$$= \Pr\Big[(\mathbf{X}'^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}} \wedge (\mathbf{X}'^w)_{\mathbf{U}'} = \mathbf{x}'_{\mathbf{U}'} \wedge (\mathbf{Y}'^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Y}'^w)_{\mathbf{V}'} = \mathbf{y}'_{\mathbf{V}'}$$
$$\wedge (\mathbf{Z}'^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}} \wedge (\mathbf{Z}'^w)_{\mathbf{W}'} = \mathbf{z}'_{\mathbf{W}'}\Big]$$
$$= \Pr\Big[(\mathbf{X}'^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}} \wedge (\mathbf{Y}'^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Z}'^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}}\Big]$$
$$\cdot \Pr\Big[(\mathbf{X}'^w)_{\mathbf{U}'} = \mathbf{x}'_{\mathbf{U}'} \wedge (\mathbf{Y}'^w)_{\mathbf{V}'} = \mathbf{y}'_{\mathbf{V}'} \wedge (\mathbf{Z}'^w)_{\mathbf{W}'} = \mathbf{z}'_{\mathbf{W}'} \Big|$$
$$(\mathbf{X}'^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}} \wedge (\mathbf{Y}'^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Z}'^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}}\Big]$$
$$> 2^{-(1-\delta)\left[\sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V_j|}} + \sum_{k=1}^{n_3}\sum_{j=1}^{K_k} \log C_k^{\underline{|W_{k,j}|}}\right]}$$
$$\cdot 2^{-(1-\delta)\left[\sum_{i=1}^{n_1} |U'_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V'_j|}} + \sum_{k=1}^{n_3}\sum_{j=1}^{K_k} \log C_k^{\underline{|W'_{k,j}|}}\right]}$$
$$= 2^{-(1-\delta)\left[\sum_{i=1}^{n_1} |U_i \cup U'_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V_j \cup V'_j|}} + \sum_{k=1}^{n_3}\sum_{j=1}^{K_k} \log C_k^{\underline{|W_{k,j} \cup W'_{k,j}|}}\right]},$$

since $(\mathbf{U}, \mathbf{V}, \mathbf{W})$ and $(\mathbf{U}', \mathbf{V}', \mathbf{W}')$ are pairwise disjoint. However, this contradicts the maximality of $(\mathbf{U}, \mathbf{V}, \mathbf{W})$.

(2) Next, we claim that

$$\sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} |V_j| \log(p_j/e) + \sum_{k=1}^{n_3}\sum_{j=1}^{K_k} |W_{k,j}| \log(C_k/e) \le S_z/\delta.$$

Let $\mathcal{B}_{\mathbf{y}'_{\mathbf{V}}}$ be the set of vectors $\mathbf{y}'_{\overline{\mathbf{V}}}$ such that $\mathbf{y}'_{\mathbf{V}}$ and $\mathbf{y}'_{\overline{\mathbf{V}}}$ are a valid injection. Similarly, let $\mathcal{C}_{\mathbf{z}'_{\mathbf{W}}}$ be the set of vectors $\mathbf{z}'_{\overline{\mathbf{W}}}$ such that $\mathbf{z}'_{\mathbf{W}}$ and $\mathbf{z}'_{\overline{\mathbf{W}}}$ are a valid IC. Since $H_\infty(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w) \ge \sum_{i=1}^{n_1} M_i \log N_i + \sum_{j=1}^{n_2} \log p_j! + \sum_{k=1}^{n_3}\sum_{j=1}^{K_k} \log C_k! - S_z$, we observe that for any $\mathbf{x}'_{\mathbf{U}}, \mathbf{y}'_{\mathbf{V}}, \mathbf{z}'_{\mathbf{W}}$,

$$\Pr\Big[(\mathbf{X}'^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}} \wedge (\mathbf{Y}'^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Z}'^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}}\Big]$$

$$= \sum_{\mathbf{x}'_{\overline{\mathbf{U}}} \in \prod_{i=1}^{n_1} [N_i]^{M_i - |U_i|}} \sum_{\mathbf{y}'_{\overline{\mathbf{V}}} \in \mathcal{B}_{\mathbf{y}'_{\mathbf{V}}}} \sum_{\mathbf{z}'_{\overline{\mathbf{W}}} \in C_{\mathbf{z}'_{\mathbf{W}}}} \Pr\Big[(\mathbf{X}^w)_{\mathbf{U}} = \mathbf{x}'_{\mathbf{U}}$$
$$\wedge (\mathbf{Y}^w)_{\mathbf{V}} = \mathbf{y}'_{\mathbf{V}} \wedge (\mathbf{Z}^w)_{\mathbf{W}} = \mathbf{z}'_{\mathbf{W}} \wedge (\mathbf{X}^w)_{\overline{\mathbf{U}}} = \mathbf{x}'_{\overline{\mathbf{U}}}$$
$$\wedge (\mathbf{Y}^w)_{\overline{\mathbf{V}}} = \mathbf{y}'_{\overline{\mathbf{V}}} \wedge (\mathbf{Z}^w)_{\overline{\mathbf{W}}} = \mathbf{z}'_{\overline{\mathbf{W}}}\Big]$$
$$\le 2^{\sum_{i=1}^{n_1} (M_i - |U_i|) \log N_i} \cdot 2^{\sum_{j=1}^{n_2} \log(p_j - |V_j|)!} \cdot 2^{\sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log(C_k - |W_{j,k}|)!}$$
$$\cdot 2^{-(\sum_{i=1}^{n_1} M_i \log N_i + \sum_{j=1}^{n_2} \log p_j! + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k! - S_z)}$$
$$= 2^{-(\sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V_j|}} + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k^{\underline{|W_{k,j}|}} - S_z)},$$

and hence,

$$H_\infty((\mathbf{X}'^w)_{\mathbf{U}}, (\mathbf{Y}'^w)_{\mathbf{V}}, (\mathbf{Z}'^w)_{\mathbf{W}})$$
$$\ge \sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V_j|}} + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k^{\underline{|W_{k,j}|}} - S_z.$$

On the other hand, because $((\mathbf{X}'^w)_{\mathbf{U}}, (\mathbf{Y}'^w)_{\mathbf{V}}, (\mathbf{Z}'^w)_{\mathbf{W}})$ is not $(1-\delta)$-dense, we have

$$H_\infty((\mathbf{X}'^w)_{\mathbf{U}}, (\mathbf{Y}'^w)_{\mathbf{V}}, (\mathbf{Z}'^w)_{\mathbf{W}})$$
$$\le (1-\delta)\left[\sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V_j|}} + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k^{\underline{|W_{k,j}|}}\right].$$

Combining those two equations together, we have

$$S_z \ge \delta\left[\sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} \log p_j^{\underline{|V_j|}} + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k^{\underline{|W_{k,j}|}}\right]$$
$$\ge \delta\left[\sum_{i=1}^{n_1} |U_i| \log N_i + \sum_{j=1}^{n_2} |V_j| \log(p_j/e) + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} |W_{k,j}| \log(C_k/e)\right],$$

where the last inequality comes from [11, Proposition 38][11]. Divided by $\delta > 0$, we get the desired result.

Hence, $(\widetilde{\mathbf{X}'}^w, \widetilde{\mathbf{Y}'}^w, \widetilde{\mathbf{Z}'}^w)$ is a $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense source such that

$$\sum_{i=1}^{n_1} P'_i \log N_i + \sum_{j=1}^{n_2} Q'_j \log(p_j/e) + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} R'_{k,j} \log(C_k/e) \le S_z/\delta.$$

Set $(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w)$ now to be $(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w)$ conditioned on $(\mathbf{X}'^w)_{\mathbf{U}} \ne \mathbf{x}'_{\mathbf{U}}$, $(\mathbf{Y}'^w)_{\mathbf{V}} \ne \mathbf{y}'_{\mathbf{V}}$, and $(\mathbf{Z}'^w)_{\mathbf{W}} \ne \mathbf{z}'_{\mathbf{W}}$ and recursively decompose $(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w)$ as long as

$$\Pr\left[\mathbf{X}^w \in \mathrm{supp}(\mathbf{X}'^w) \wedge \mathbf{Y}^w \in \mathrm{supp}(\mathbf{Y}'^w) \wedge \mathbf{Z}^w \in \mathrm{supp}(\mathbf{Z}'^w)\right] > \gamma.$$

Observe that

$$H_\infty(\mathbf{X}'^w, \mathbf{Y}'^w, \mathbf{Z}'^w)$$
$$\ge \sum_{i=1}^{n_1} M_i \log N_i + \sum_{j=1}^{n_2} \log p_j! + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k! - (S_z + \log \gamma^{-1})$$

at any point in this decomposition process since

$$\Pr\left[\mathbf{X}'^w = \mathbf{x}' \wedge \mathbf{Y}'^w = \mathbf{y}' \wedge \mathbf{Z}'^w = \mathbf{z}'\right]$$
$$= \Pr\Big[\mathbf{X}^w = \mathbf{x}' \wedge \mathbf{Y}^w = \mathbf{y}' \wedge \mathbf{Z}^w = \mathbf{z}' \Big|$$
$$\mathbf{X}^w \in \mathrm{supp}(\mathbf{X}'^w) \wedge \mathbf{Y}^w \in \mathrm{supp}(\mathbf{Y}'^w) \wedge \mathbf{Z}^w \in \mathrm{supp}(\mathbf{Z}'^w)\Big]$$
$$\le \frac{\Pr[\mathbf{X}^w = \mathbf{x}' \wedge \mathbf{Y}^w = \mathbf{y}' \wedge \mathbf{Z}^w = \mathbf{z}']}{\Pr[\mathbf{X}^w \in \mathrm{supp}(\mathbf{X}'^w) \wedge \mathbf{Y}^w \in \mathrm{supp}(\mathbf{Y}'^w) \wedge \mathbf{Z}^w \in \mathrm{supp}(\mathbf{Z}'^w)]}$$
$$\le \frac{2^{-(\sum_{i=1}^{n_1} M_i \log N_i + \sum_{j=1}^{n_2} \log p_j! + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k! - S_z)}}{\gamma}$$
$$= 2^{-(\sum_{i=1}^{n_1} M_i \log N_i + \sum_{j=1}^{n_2} \log p_j! + \sum_{k=1}^{n_3} \sum_{j=1}^{K_k} \log C_k! - (S_z + \log \gamma^{-1}))}.$$

Note that $\mathrm{supp}(\mathbf{X}'^w), \mathrm{supp}(\mathbf{Y}'^w), \mathrm{supp}(\mathbf{Z}'^w)$ decreases in every step, and since $\mathrm{supp}(\mathbf{X}^w), \mathrm{supp}(\mathbf{Y}^w), \mathrm{supp}(\mathbf{Z}^w)$ are finite, after finitely many steps, this process ends with $(\mathbf{X}'^w_{\mathsf{final}}, \mathbf{Y}'^w_{\mathsf{final}}, \mathbf{Z}'^w_{\mathsf{final}})$ with $\Pr\left[\mathbf{X}^w \in \mathrm{supp}(\mathbf{X}'^w_{\mathsf{final}}) \wedge \mathbf{Y}^w \in \mathrm{supp}(\mathbf{Y}'^w_{\mathsf{final}}) \wedge \mathbf{Z}^w \in \mathrm{supp}(\mathbf{Z}'^w_{\mathsf{final}})\right] > \gamma$. Hence, $(\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$ is a convex combination of finitely many $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense sources and $(\mathbf{X}'^w_{\mathsf{final}}, \mathbf{Y}'^w_{\mathsf{final}}, \mathbf{Z}'^w_{\mathsf{final}})$, where $\mathbf{P}', \mathbf{Q}', \mathbf{R}'$ satisfy the inequality

$$\sum_{i=1}^{n_1} P'_i \log N_i + \sum_{i=1}^{n_2} Q'_i \log(p_i/e) + \sum_{i=1}^{n_3} \sum_{j=1}^{K_i} R'_{i,j} \log(C_i/e) \le \frac{S_z + \log \gamma^{-1}}{\delta}.$$

This implies that $(\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$ is γ-close to a convex combination of $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense sources where $\sum_{i=1}^{n_1} P'_i \log N_i + \sum_{i=1}^{n_2} Q'_i \log(p_i/e) + \sum_{i=1}^{n_3} \sum_{j=1}^{K_i} R'_{i,j} \log(C_i/e) \le (S_z + \log \gamma^{-1})/\delta$. □

Let $(\widetilde{\mathbf{X}}^w, \widetilde{\mathbf{Y}}^w, \widetilde{\mathbf{Z}}^w)$ be the convex combination of $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense sources that is γ-close to $(\mathbf{X}^w, \mathbf{Y}^w, \mathbf{Z}^w)$ for a $\delta = \delta_z$ to be determined later. For every $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$ source $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}, \widetilde{\mathbf{Z}})$ in said convex combination, let $(\widetilde{\mathbf{X}'}, \widetilde{\mathbf{Y}'}, \widetilde{\mathbf{Z}'})$ be the corresponding $(\mathbf{P}', \mathbf{Q}', \mathbf{R}')$-bit-fixing source, i.e., $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}, \widetilde{\mathbf{Z}})$ and $(\widetilde{\mathbf{X}'}, \widetilde{\mathbf{Y}'}, \widetilde{\mathbf{Z}'})$ are fixed on the same coordinates to the same values. The following claim bounds the distinguishing advantage between $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}, \widetilde{\mathbf{Z}})$ and $(\widetilde{\mathbf{X}'}, \widetilde{\mathbf{Y}'}, \widetilde{\mathbf{Z}'})$ for any $(\mathbf{S}, \mathbf{T}, \mathbf{L})$-query distinguisher.

Claim 5. *For any $(\mathbf{P}', \mathbf{Q}', \mathbf{R}', 1-\delta)$-dense source $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}, \widetilde{\mathbf{Z}})$ and its corresponding $(\mathbf{P}', \mathbf{Q}', \mathbf{R}')$-bit-fixing source $(\widetilde{\mathbf{X}'}, \widetilde{\mathbf{Y}'}, \widetilde{\mathbf{Z}'})$, it holds that for any (adaptive) distinguisher $\mathcal{D}$ that queries at most $\mathbf{S} = (S_1, \ldots, S_{n_1})$ coordinates of each oracle that corresponds to $\widetilde{\mathbf{X}}$ where $S_i < P'_i$ for each $i \in [n_1]$, $\mathbf{T} = (T_1, \ldots, T_{n_2})$ coordinates of each oracle that corresponds to $\widetilde{\mathbf{Y}}$ where $T_i < Q'_i$ for each $i \in [n_2]$, and $\mathbf{L} = (\mathbf{L}_1, \ldots, \mathbf{L}_{n_3})$ coordinates of each oracle that corresponds to $\widetilde{\mathbf{Z}}$ where $\mathbf{L}_k = (L_{k,1}, \ldots, L_{k,K_k})$ and $L_{i,j} < R'_{i,j}$ for each $j \in [K_i]$ and $i \in [n_3]$,*

$$\left|\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}, \widetilde{\mathbf{Z}}} = 1\right] - \Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}'}, \widetilde{\mathbf{Y}'}, \widetilde{\mathbf{Z}'}} = 1\right]\right|$$
$$\le \delta\left(\sum_{i=1}^{n_1} S_i \log N_i + \sum_{i=1}^{n_2} T_i \log p_i + \sum_{i=1}^{n_3} \sum_{j=1}^{K_i} L_{i,j} \log C_i\right),$$

and

$$\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}, \widetilde{\mathbf{Z}}} = 1\right]$$
$$\le 2^{\delta\left(\sum_{i=1}^{n_1} S_i \log N_i + \sum_{i=1}^{n_2} T_i \log p_i + \sum_{i=1}^{n_3} \sum_{j=1}^{K_i} L_{i,j} \log C_i\right)}$$
$$\cdot \Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}'}, \widetilde{\mathbf{Y}'}, \widetilde{\mathbf{Z}'}} = 1\right].$$

[11] [11, Proposition 38] says that $N^{\underline{j}} \ge (N/e)^j$, where $a^{\underline{b}} \coloneqq a!/(a-b)!$.

Proof of Claim 5. Without loss of generality, assume that $\mathcal{D}$ is deterministic and does not query any of the fixed positions, make the same query twice, or make an inverse query after making the corresponding forward query or vice-versa. Let $T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}$ and $T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}$ be the random variables corresponding to the transcripts containing the query/answer pairs resulting from $\mathcal{D}$'s interaction with $(\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}})$ and $(\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'})$, respectively.

For a fixed transcript t, denote by $\mathrm{p}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}(t)$ and $\mathrm{p}_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}(t)$ the probabilities that $(\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}})$ and $(\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'})$, respectively, produce the answers in t if the queries in t are asked. Since $\mathcal{D}$ is deterministic, $\Pr\left[T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=t\right]\in\left\{0,\mathrm{p}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}(t)\right\}$, and similarly, $\Pr\left[T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=t\right]\in\left\{0,\mathrm{p}_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}(t)\right\}$. Denote by $\mathcal{T}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}$ the set of all transcripts t for which $\Pr\left[T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=t\right]>0$. For such t, $\Pr\left[T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=t\right]=\mathrm{p}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}(t)$ and also $\Pr\left[T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=t\right]=\mathrm{p}_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}(t)$. Observe that for every transcript t,

$$\begin{aligned}&\mathrm{p}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}(t)\\&\quad\le 2^{-(1-\delta)\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2}\log(p_i-Q'_i)^{\underline{T_i}}+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i}\log(C_i-R'_{i,j})^{\underline{L_{i,j}}}\right]},\end{aligned}\tag{7}$$

and

$$\begin{aligned}&\mathrm{p}_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}(t)\\&\quad\le 2^{-\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2}\log(p_i-Q'_i)^{\underline{T_i}}+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i}\log(C_i-R'_{i,j})^{\underline{L_{i,j}}}\right]},\end{aligned}\tag{8}$$

as $(\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}})$ is $(\mathbf{P}',\mathbf{Q}',\mathbf{R}',1-\delta)$-dense and $(\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'})$ is $(\mathbf{P}',\mathbf{Q}',\mathbf{R}')$-fixed.

Towards proving the first part of the claim, observe that $\mathcal{D}$'s output can be computed from the transcript (including whether a query was a forward for an inverse query) by just running $\mathcal{D}$ and providing the answers to its queries from the transcript. Hence,

$$\begin{aligned}&\left|\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=1\right]-\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=1\right]\right|\le \mathrm{SD}\left(T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}},T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}\right)\\&\quad=\sum_t\max\left\{0,\Pr\left[T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=t\right]-\Pr\left[T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=t\right]\right\}\\&\quad=\sum_{t\in\mathcal{T}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}}\max\left\{0,\mathrm{p}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}(t)-\mathrm{p}_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}(t)\right\}\\&\quad=\sum_{t\in\mathcal{T}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}}\mathrm{p}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}(t)\cdot\max\left\{0,1-\frac{\mathrm{p}_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}(t)}{\mathrm{p}_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}(t)}\right\}\\&\quad\le 1-2^{-\delta\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2}\log(p_i-Q'_i)^{\underline{T_i}}+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i}\log(C_i-R'_{i,j})^{\underline{L_{i,j}}}\right]}\\&\quad\le 1-2^{-\delta\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2} T_i\log p_i+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j}\log C_i\right]}\\&\quad\le -\delta\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2} T_i\log p_i+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j}\log C_i\right],\end{aligned}$$

where the first sum is over all possible transcripts and where the last inequality uses $2^{-x}\ge 1-x$ for $x\ge 0$ and $a^{\underline{b}}\le a^b$ for $a,b\in\mathbb{N}$.

As for the second part of the claim, observe that due to the equation (7),(8), and the support of $T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}$ being a subset of $T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}$,

$$\begin{aligned}&\Pr\left[T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=t\right]\\&\le 2^{\delta\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2}\log(p_i-Q'_i)^{\underline{T_i}}+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i}\log(C_i-R'_{i,j})^{\underline{L_{i,j}}}\right]}\\&\qquad\cdot\Pr\left[T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=t\right]\\&\le 2^{\delta\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2} T_i\log p_i+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j}\log C_i\right]}\cdot\Pr\left[T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=t\right],\end{aligned}$$

for any transcript t. Let $\mathcal{T}_{\mathcal{D}}$ be the set of transcripts where $\mathcal{D}$ outputs 1. Then we have

$$\begin{aligned}&\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=1\right]=\sum_{t\in\mathcal{T}_{\mathcal{D}}}\Pr\left[T_{\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}}}=t\right]\\&\le 2^{\delta\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2} T_i\log p_i+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j}\log C_i\right]}\\&\qquad\cdot\sum_{t\in\mathcal{T}_{\mathcal{D}}}\Pr\left[T_{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=t\right]\\&=2^{\delta\left[\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2} T_i\log p_i+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j}\log C_i\right]}\cdot\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'}}=1\right],\end{aligned}$$

which concludes the proof. □

Let $(\widetilde{\mathbf{X}'}^w,\widetilde{\mathbf{Y}'}^w,\widetilde{\mathbf{Z}'}^w)$ be obtained by replacing $(\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}},\widetilde{\mathbf{Z}})$ by $(\widetilde{\mathbf{X}'},\widetilde{\mathbf{Y}'},\widetilde{\mathbf{Z}'})$ in $(\widetilde{\mathbf{X}}^w,\widetilde{\mathbf{Y}}^w,\widetilde{\mathbf{Z}}^w)$. Setting

$$\delta_z=\frac{S_z+\log\gamma^{-1}}{\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2} T_i\log p_i+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j}\log C_i},$$

Claim 4 and Claim 5 imply

$$\left|\Pr\left[\mathcal{D}^{\mathbf{X}^w,\mathbf{Y}^w,\mathbf{Z}^w}(z)=1\right]-\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}'}^w,\widetilde{\mathbf{Y}'}^w,\widetilde{\mathbf{Z}'}^w}(z)=1\right]\right|\le\Lambda+\gamma,\tag{9}$$

as well as

$$\Pr\left[\mathcal{D}^{\mathbf{X}^w,\mathbf{Y}^w,\mathbf{Z}^w}(z)=1\right]\le 2^{\Lambda}\cdot\Pr\left[\mathcal{D}^{\widetilde{\mathbf{X}'}^w,\widetilde{\mathbf{Y}'}^w,\widetilde{\mathbf{Z}'}^w}(z)=1\right]+\gamma,$$

where

$$\Lambda=\frac{(S+\log\gamma^{-1})\left(\sum_{i=1}^{n_1} S_i\log N_i+\sum_{i=1}^{n_2} T_i\log p_i+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} L_{i,j}\log C_i\right)}{\sum_{i=1}^{n_1} P_i\log N_i+\sum_{i=1}^{n_2} Q_i\log(p_i/e)+\sum_{i=1}^{n_3}\sum_{j=1}^{K_i} R_{i,j}\log(C_i/e)}.$$

Moreover, note that for the above choice of δ_z, we have $(\mathbf{P}',\mathbf{Q}',\mathbf{R}')=(\mathbf{P},\mathbf{Q},\mathbf{R})$, i.e., the sources $(\widetilde{\mathbf{X}'}^w,\widetilde{\mathbf{Y}'}^w,\widetilde{\mathbf{Z}'}^w)$ are $(\mathbf{P},\mathbf{Q},\mathbf{R})$-fixed, as desired.

Claim 6. $\mathbb{E}_z[S_z]\le S$ *and* $\Pr[S_{f(\mathbf{X},\mathbf{Y},\mathbf{Z})}>S+\log\gamma^{-1}]\le\gamma$.

Proof of Claim 6. Observe that $H_\infty(\mathbf{X}^w,\mathbf{Y}^w,\mathbf{Z}^w)=H_\infty((\mathbf{X},\mathbf{Y},\mathbf{Z})|W=w)=H((\mathbf{X},\mathbf{Y},\mathbf{Z})|W=w)$ since, conditioned on $W=w$, $(\mathbf{X},\mathbf{Y},\mathbf{Z})$ is distributed uniformly over all values $(\mathbf{x},\mathbf{y},\mathbf{z})$ with $f(\mathbf{x},\mathbf{y},\mathbf{z})=w$. Hence,

$$\begin{aligned}\mathbb{E}_z[S_z]&=\sum_{i=1}^{n_1}M_i\log N_i+\sum_{i=1}^{n_2}\log p_i!+\sum_{i=1}^{n_3}K_i\log C_i!-\mathbb{E}_z\left[H_\infty((\mathbf{X},\mathbf{Y},\mathbf{Z})|W=w)\right]\\&=\sum_{i=1}^{n_1}M_i\log N_i+\sum_{i=1}^{n_2}\log p_i!+\sum_{i=1}^{n_3}K_i\log C_i!-\mathbb{E}_z\left[H((\mathbf{X},\mathbf{Y},\mathbf{Z})|W=w)\right]\\&=\sum_{i=1}^{n_1}M_i\log N_i+\sum_{i=1}^{n_2}\log p_i!+\sum_{i=1}^{n_3}K_i\log C_i!-H((\mathbf{X},\mathbf{Y},\mathbf{Z})|W=w)\le S.\end{aligned}$$

Again, due to the uniformity of $(\mathbf{X},\mathbf{Y},\mathbf{Z})$, $\Pr[f(\mathbf{X},\mathbf{Y},\mathbf{Z})=z]=2^{-S_z}$. Hence,

$$\begin{aligned}&\Pr[S_{f(\mathbf{X},\mathbf{Y},\mathbf{Z})}>S+\log\gamma^{-1}]\\&\qquad=\sum_{w\in\{0,1\}^S:S_z>S+\log\gamma^{-1}}\Pr[f(\mathbf{X},\mathbf{Y},\mathbf{Z})=w]\end{aligned}$$

$$\leq 2^S \cdot 2^{-(S+\log \gamma^{-1})} \leq \gamma. \qquad \square$$

Now the lemma follows (using $(\mathbf{X'}^w, \mathbf{Y'}^w, \mathbf{Z'}^w) \coloneqq (\widetilde{\mathbf{X'}}^w, \widetilde{\mathbf{Y'}}^w, \widetilde{\mathbf{Z'}}^w)$) by taking expectations over w of the equation (9) and applying the first part of Claim 6. $\square$

B.2 From the Bit-Fixing Model to the Auxiliary-Input Model

Capturing the Models. A tuple of oracles $\left(\{O_{1,i}\}_{i=1}^{n_1}, \{O_{2,j}\}_{j=1}^{n_2}, \{O_{3,k}\}_{k=1}^{n_3}\right)$ has two interfaces $(\{O_{1,i}.\mathsf{pre}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{pre}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{pre}\}_{k=1}^{n_3})$ that is accessible during a preprocessing phase, and $(\{O_{1,i}.\mathsf{on}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{on}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{on}\}_{k=1}^{n_3})$ that is accessible during an online phase, where $\left(\{O_{1,i}.\mathsf{pre}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{pre}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{pre}\}_{k=1}^{n_3}\right)$ is accessible only once before any calls to $\left(\{O_{1,i}.\mathsf{on}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{on}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{on}\}_{k=1}^{n_3}\right)$ are made. We consider the oracles as follows:

- **Random Oracle + Generic Group Oracle + Ideal Cipher** $\mathsf{RO{+}GG{+}IC}(\{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$**:** Samples random oracle function tables $\mathsf{H}_i \leftarrow \mathcal{H}_{M_i,N_i}$ for each $i \in [n_1]$, where $\mathcal{H}_{M_i,N_i}$ is the set of all functions from $[M_i] \to [N_i]$, samples random injections $\tau_j \leftarrow \mathcal{I}_{p_j,\ell_j}$ for each $j \in [n_2]$, where $\mathcal{I}_{p_j,\ell_j}$ is the set of all injections from $\mathbb{Z}_{p_j} \to \mathbb{G}_j$ where $\mathbb{G}_j$ is the set of bitstrings of length $\ell_j \geq \log p_j$, and samples random permutations $\pi_s \leftarrow \mathcal{P}_{C_k}$ for each $s \in [K_k]$, where $\mathcal{P}_{C_k}$ is the set of all permutations from $[C_k] \to [C_k]$; offers no functionality at $(\{O_{1,i}.\mathsf{pre}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{pre}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{pre}\}_{k=1}^{n_3})$; for the online queries,
  - for each $i \in [n_1]$, answers queries $x \in [M_i]$ to H_i at $O_{1,i}.\mathsf{on}$ by the corresponding value $\mathsf{H}_i(x) \in [N_i]$,
  - for each $j \in [n_2]$, answers *forward* queries $x' \in \mathbb{Z}_{p_j}$ to τ_j at $O_{2,j}.\mathsf{on}$ by the corresponding value $\tau_j(x') \in \mathbb{G}_j$; answers *group-operation* queries (s, s') at $O_{2,j}.\mathsf{on}$ as follows: if $s = \tau_j(x)$ and $s' = \tau_j(x')$ for some x, x', the oracle replies by $\tau_j(x{+}x')$ and by $\perp$ otherwise; answers *inverse* queries s at $O_{2,j}.\mathsf{on}$ by returning $\tau_j^{-1}(s)$ if s is in the range of τ_j and $\perp$ otherwise, and
  - for each $k \in [n_3]$, answers both *forward* and *backward* queries $(s, x) \in [K_k] \times [C_k]$ at $O_{3,k}.\mathsf{on}$ by the corresponding value $\pi_s(x) \in [C_k]$ or $\pi_s^{-1}(x) \in [C_k]$, respectively.
- **Auxiliary-Input Random Oracle + Generic Group Oracle + Ideal Cipher** $\mathsf{AI\text{-}RO{+}GG{+}IC}(\{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$**:** Samples $\mathsf{H}_i, \tau_j, \pi_s$ for each $i \in [n_1]$, $j \in [n_2]$, and $s \in [K_k]$ where $k \in [n_3]$ as explained in $\mathsf{RO{+}GG{+}IC}(\{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$ above; outputs all of H_i, τ_j, and π_s at $(\{O_{1,i}.\mathsf{pre}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{pre}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{pre}\}_{k=1}^{n_3})$; $(\{O_{1,i}.\mathsf{on}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{on}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{on}\}_{k=1}^{n_3})$ behaves the same as $\mathsf{RO{+}GG{+}IC}(\{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$. When the parameters $(\{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$ are clear in context, we sometimes abuse the notation and simply say $\mathsf{AI\text{-}RO{+}GG{+}IC}$.
- **Bit-Fixing Random Oracle + Generic Group Oracle + Ideal Cipher** $\mathsf{BF\text{-}RO{+}GG{+}IC}(\mathbf{P}, \mathbf{Q}, \mathbf{R}, \{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$**:** Samples a random oracle function table $\mathsf{H}_i \leftarrow \mathcal{H}_{M_i,N_i}$ at $O_{1,i}.\mathsf{pre}$ for each $i \in [n_1]$, where $\mathcal{H}_{M_i,N_i}$ is the set of all functions from $[M_i] \to [N_i]$, samples a random size-p_j subset $\mathcal{Y}_j$ of $\mathbb{G}_j$ for each $j \in [n_2]$ and outputs $\mathcal{Y}_j$ at $O_{2,j}.\mathsf{pre}$, and samples a random permutation $\pi_s \leftarrow \mathcal{P}_{C_k}$ at $O_{3,k}.\mathsf{pre}$ for each $s \in [K_k]$, where $\mathcal{P}_{C_k}$ is the set of all permutations from $[C_k] \to [C_k]$.
  - Given $\mathbf{P} = (P_1, \ldots, P_{n_1}) \in \mathbb{Z}_{M_1} \times \cdots \times \mathbb{Z}_{M_{n_1}}$, for each $i \in [n_1]$, takes a list at $O_{1,i}.\mathsf{pre}$ of at most P_i query/answer pairs that override H_i in the corresponding positions,
  - given $\mathbf{Q} = (Q_1, \ldots, Q_{n_2}) \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_{n_1}}$, for each $j \in [n_2]$, takes a list at $O_{2,j}.\mathsf{pre}$ of at most Q_j query/answer pairs without collisions and all answers in $\mathcal{Y}_j$; samples a random injection $\tau_j \leftarrow \mathcal{I}_{p_j,\ell_j}$ with range $\mathcal{Y}_j$ and consistent with said list, and
  - given $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_{n_3}) \in (\mathbb{Z}_{C_1})^{K_1} \times \cdots \times (\mathbb{Z}_{C_{n_3}})^{K_{n_3}}$ where $\mathbf{R}_k = (R_{k,1}, \ldots, R_{k,K_k})$ for each $k \in [n_3]$, takes a list at $O_{3,k}.\mathsf{pre}$ of at most $R_{k,s}$ query/answer pairs (without collisions for each $s \in [K_k]$); samples a random permutation $\pi_s \leftarrow \mathcal{P}_{C_k}$ consistent with said list for each $s \in [K_k]$.
  - $(\{O_{1,i}.\mathsf{on}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{on}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{on}\}_{k=1}^{n_3})$ behaves the same as $\mathsf{RO{+}GG{+}IC}(\{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$. When the parameters $(\{M_i, N_i\}_{i=1}^{n_1}, \{p_j, \ell_j\}_{j=1}^{n_2}, \{K_k, C_k\}_{k=1}^{n_3})$ are clear in context, we sometimes abuse the notation and simply say $\mathsf{BF\text{-}RO{+}GG{+}IC}(\mathbf{P}, \mathbf{Q}, \mathbf{R})$.

Attackers with Oracle-Dependent Advice. We define the attackers $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$ similar to that of prior work [11, 12], which consist of a preprocessing attacker $\mathcal{A}_{\mathsf{pre}}$ and an online attacker $\mathcal{A}_{\mathsf{on}}$, which carries out the actual attack using the output of $\mathcal{A}_{\mathsf{pre}}$. The difference is that we consider a tuple of oracles instead of a single oracle. More precisely, in the presence of a tuple of oracles $\left(\{O_{1,i}\}_{i=1}^{n_1}, \{O_{2,j}\}_{j=1}^{n_2}, \{O_{3,k}\}_{k=1}^{n_3}\right)$, $\mathcal{A}_{\mathsf{pre}}$ interacts with $(\{O_{1,i}.\mathsf{pre}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{pre}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{pre}\}_{k=1}^{n_3})$ and $\mathcal{A}_{\mathsf{on}}$ with $\left(\{O_{1,i}.\mathsf{on}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{on}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{on}\}_{k=1}^{n_3}\right)$.

Definition B.3. Given $\mathbf{S} = (S_1, \ldots, S_{n_1}), \mathbf{T} = (T_1, \ldots, T_{n_2})$, and $\mathbf{L} = (\mathbf{L}_1, \ldots, \mathbf{L}_{n_3})$ where $\mathbf{L}_k = (L_{k,1}, \ldots, L_{k,K_k})$ for $k \in [n_3]$, an $(S, \mathbf{S}, \mathbf{T}, \mathbf{L})$*-attacker* $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$ *in the* $\left(\{O_{1,i}\}_{i=1}^{n_1}, \{O_{2,j}\}_{j=1}^{n_2}, \{O_{3,k}\}_{k=1}^{n_3}\right)$*-model* consists of two procedures

- $\mathcal{A}_{\mathsf{pre}}$, which is computationally unbounded, interacts with $(\{O_{1,i}.\mathsf{pre}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{pre}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{pre}\}_{k=1}^{n_3})$, and outputs an S-bit string as hint, and
- $\mathcal{A}_{\mathsf{on}}$, which takes an S-bit auxiliary input and makes at most $(\mathbf{S}, \mathbf{T}, \mathbf{L})$ queries to $(\{O_{1,i}.\mathsf{on}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{on}\}_{j=1}^{n_2}, \{O_{3,k}.\mathsf{on}\}_{k=1}^{n_3})$. That is, $\mathcal{A}_{\mathsf{on}}$ makes at most S_i queries to $O_{1,i}.\mathsf{on}$ for each $i \in [n_1]$, T_j queries to $O_{2,j}.\mathsf{on}$ for each $j \in [n_2]$, and $L_{k,s}$ queries to $O_{3,k}.\mathsf{on}$ for each $k \in [n_3]$ and for each key $s \in [K_k]$.

The definition of an *application* G and the *success* of $\mathcal{A}$ on G (see Definition A.3) can be easily extended by defining $O \coloneqq \left(\{O_{1,i}\}_{i=1}^{n_1}, \{O_{2,j}\}_{j=1}^{n_2}, \{O_{3,k}\}_{k=1}^{n_3}\right)$, $O.\mathsf{pre} \coloneqq \left(\{O_{1,i}.\mathsf{pre}\}_{i=1}^{n_1}, \{O_{2,j}.\mathsf{pre}\}_{j=1}^{n_2},\right.$

$\{O_{3,k}.\mathrm{pre}\}_{k=1}^{n_3}\Big)$, and $O.\mathrm{on} \coloneqq (\{O_{1,i}.\mathrm{on}\}_{i=1}^{n_1}, \{O_{2,j}.\mathrm{on}\}_{j=1}^{n_2}, \{O_{3,k}.\mathrm{on}\}_{k=1}^{n_3})$. The *advantage* of an attacker $\mathcal{A}$ for an *indistinguishability/unpredictability* application G can also be defined accordingly for $O = \Big(\{O_{1,i}\}_{i=1}^{n_1}, \{O_{2,j}\}_{j=1}^{n_2}, \{O_{3,k}\}_{k=1}^{n_3}\Big)$ as defined in Definition A.4.

Additive Error for Arbitrary Applications in the ROM+GGM+ICM. Using Lemma B.2, one can extend [12, Theorem 5] and [11, Theorem 1] to the translation from bit-fixing model with multiple idealized models (ROM+GGM+ICM) to the corresponding auxiliary-input model at the cost of an additive term which is similar to those from [12, Theorem 5] and [11, Theorem 1].

Theorem B.4. *For every $\gamma > 0$, if an application G is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon')$-secure in the* BF-RO+GG+IC$(\mathbf{P}, \mathbf{Q}, \mathbf{R}, \{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model for any $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ as introduced in Definition B.1 which satisfies*

$$\sum_{i=1}^{n_1} P_i \log N_i + \sum_{i=1}^{n_2} Q_i \log(p_i/e) + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i} R_{i,j}\log(C_i/e) \le \eta,$$

then it is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon)$-secure in the AI-RO+GG+IC$(\{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model, for $\varepsilon \le \varepsilon' + \Lambda + 2\gamma$, where*

$$\Lambda = \frac{2(S+\log\gamma^{-1})\left(\sum_{i=1}^{n_1}(S_i)_G^{\mathrm{comb}}\log N_i + \sum_{i=1}^{n_2} 3(T_i)_G^{\mathrm{comb}}\log p_i + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i} 2(L_{i,j})_G^{\mathrm{comb}}\log C_i\right)}{\eta},$$

e is the Euler constant, and $(S_i)_G^{\mathrm{comb}}$, $(T_i)_G^{\mathrm{comb}}$ and $(L_{i,j})_G^{\mathrm{comb}}$ are the combined query complexity corresponding to G that corresponds to the random oracles, the generic group oracles, and the ideal ciphers, respectively.

Proof. Fix $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ as well as γ. Let G be an arbitrary application and C be the corresponding challenger. Moreover, fix an $(S, \mathbf{S}, \mathbf{T}, \mathbf{L})$-attacker $\mathcal{A} = (\mathcal{A}_{\mathrm{pre}}, \mathcal{A}_{\mathrm{on}})$, and let $\{(\mathbf{X}', \mathbf{Y}', \mathbf{Z}')^{w,\mathcal{Y}}\}$, indexed by $w \in \{0,1\}^S$ and $\mathcal{Y} = (\mathcal{Y}_1, \dots, \mathcal{Y}_{n_2})$ where $\mathcal{Y}_i$ is a size-p_i subset of $\mathbb{G}_i$ for each $i \in [n_2]$, be the family of distributions guaranteed to exist by Lemma B.2. Consider the following $(S, \mathbf{S}, \mathbf{T}, \mathbf{L})$-attacker $\mathcal{A}' = (\mathcal{A}'_{\mathrm{pre}}, \mathcal{A}'_{\mathrm{on}})$ that is expected to interact with BF-RO+GG+IC:

- $\mathcal{A}'_{\mathrm{pre}}$ obtains the set $\mathcal{Y}$ from $\{\text{BF-GG.pre}\}_{i=1}^{n_2}$ and internally simulates $\mathcal{A}_{\mathrm{pre}}$ on a uniformly random input $\mathbf{X} \in \prod_{i=1}^{n_1}[M_i]$, a uniformly random injection $\mathbf{Y} \in \prod_{j=1}^{n_2}\mathbb{Z}_{p_j}$ with range $\mathcal{Y}$, a uniformly random IC $\mathbf{Z} \in \prod_{k=1}^{n_3}[K_k] \times [C_k]$ to obtain $w \leftarrow \mathcal{A}_{\mathrm{pre}}^{\text{AI-RO+GG+IC.pre}}$. Then, it samples one of the $(\mathbf{P}, \mathbf{Q}, \mathbf{R})$-bit-fixing sources $(\mathbf{X}'', \mathbf{Y}'', \mathbf{Z}'')$ making up $(\mathbf{X}', \mathbf{Y}', \mathbf{Z}')^{w,\mathcal{Y}}$ and presets BF-RO+GG+IC to match $(\mathbf{X}'', \mathbf{Y}'', \mathbf{Z}'')$ on the at most $(\mathbf{P}, \mathbf{Q}, \mathbf{R})$ points (pairwise and elementwise) where $(\mathbf{X}'', \mathbf{Y}'', \mathbf{Z}'')$ is fixed. The output of $\mathcal{A}'_{\mathrm{pre}}$ is w.
- $\mathcal{A}'_{\mathrm{on}}$ works exactly the same as $\mathcal{A}_{\mathrm{on}}$.

Let $\mathcal{D}$ be a distinguisher — making forward, group-operation, inverse, and backward queries to an (n_1, n_2, n_3)-source — that internally runs the combination of $\mathcal{A}_{\mathrm{on}} = \mathcal{A}'_{\mathrm{on}}$ and C. It answers their queries as follows:

(1) It passes forward and backward queries to and back from its own oracle.
(2) It answers group-operation queries (s, s') (to the generic group oracle) by making two backward queries to its own oracle for s and s', obtaining i and j, respectively, making a forward query $i + j$, and passing the answer to $\mathcal{A}_{\mathrm{on}}$ or C (unless one of the answers to the backward queries was $\perp$, in which case $\perp$ is returned).

Note that $\mathcal{D}^{\mathbf{X},\mathbf{Y},\mathbf{Z}}(f(\mathbf{X}, \mathbf{Y}, \mathbf{Z}))$ is identical to

$$\mathcal{A}_{\mathrm{on}}^{\text{AI-RO+GG+IC.on}}(\mathcal{A}_{\mathrm{pre}}^{\text{AI-RO+GG+IC.pre}}) \leftrightarrow \mathrm{C}^{\text{AI-RO+GG+IC.on}},$$

and $\mathcal{D}^{(\mathbf{X}', \mathbf{Y}', \mathbf{Z}')^{f(\mathbf{X},\mathbf{Y},\mathbf{Z}),\mathrm{im}(\mathbf{Y})}}(f(\mathbf{X}, \mathbf{Y}, \mathbf{Z}))$ is identical to

$$\mathcal{A}'^{\text{BF-RO+GG+IC.on}}_{\mathrm{on}}(\mathcal{A}'^{\text{BF-RO+GG+IC.pre}}_{\mathrm{pre}}) \leftrightarrow \mathrm{C}^{\text{BF-RO+GG+IC.on}}.$$

Furthermore, $\mathcal{D}$ is a distinguisher taking an S-bit input and making at most $(S_i)_G^{\mathrm{comb}}$ queries to the i^{th} random oracle for $i \in [n_1]$, $3(T_j)_G^{\mathrm{comb}}$ queries to the j^{th} generic group oracle for $j \in [n_2]$, and $2(L_{k,j})_G^{\mathrm{comb}}$ queries to the k^{th} ideal cipher for $k \in [n_3]$ and for each key $j \in [K_k]$. Therefore, by Lemma B.2,

$$\begin{aligned}&\mathrm{Succ}_{G,\text{AI-RO+GG+IC}}(\mathcal{A})\\ &\le \mathrm{Succ}_{G,\text{BF-RO+GG+IC}}(\mathcal{A}')\\ &+ \frac{(S+\log\gamma^{-1})\left(\sum_{i=1}^{n_1}(S_i)_G^{\mathrm{comb}}\log N_i + \sum_{i=1}^{n_2}3(T_j)_G^{\mathrm{comb}}\log p_i + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i}2(L_{k,j})_G^{\mathrm{comb}}\log C_i\right)}{\eta}\\ &+\gamma,\end{aligned}$$

for any $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ satisfying

$$\sum_{i=1}^{n_1} P_i \log N_i + \sum_{i=1}^{n_2} Q_i \log(p_i/e) + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i} R_{i,j}\log(C_i/e) \le \eta.$$

Since there is only an additive term between the two success probabilities, the above inequality implies

$$\begin{aligned}&\mathrm{Adv}_{G,\text{AI-RO+GG+IC}}(\mathcal{A})\\ &\le \mathrm{Adv}_{G,\text{BF-RO+GG+IC}}(\mathcal{A}')\\ &+ \frac{(S+\log\gamma^{-1})\left(\sum_{i=1}^{n_1}(S_i)_G^{\mathrm{comb}}\log N_i + \sum_{i=1}^{n_2}3(T_j)_G^{\mathrm{comb}}\log p_i + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i}2(L_{k,j})_G^{\mathrm{comb}}\log C_i\right)}{\eta}\\ &+2\gamma,\end{aligned}$$

for both indistinguishability and unpredictability applications. Note that the extra factor of 2 is technically only necessary for indistinguishability applications [12]. □

Multiplicative Error for Unpredictability Applications in the ROM+GGM+ICM.

Theorem B.5. *For every $\gamma > 0$, if an* unpredictability *application G is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon')$-secure in the* BF-RO+GG+IC$(\mathbf{P}, \mathbf{Q}, \mathbf{R}, \{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model for any $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ as introduced in Definition B.1 satisfying*

$$\begin{aligned}&(S+\log\gamma^{-1})\Bigg(\sum_{i=1}^{n_1}(S_i)_G^{\mathrm{comb}}\log N_i + \sum_{i=1}^{n_2}3(T_i)_G^{\mathrm{comb}}\log p_i\\ &\qquad + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i}2(L_{i,j})_G^{\mathrm{comb}}\log C_i\Bigg)\\ &\le \sum_{i=1}^{n_1} P_i\log N_i + \sum_{i=1}^{n_2} Q_i\log(p_i/e) + \sum_{i=1}^{n_3}\sum_{j=1}^{K_i} R_{i,j}\log(C_i/e),\end{aligned}$$

then it is $(S, \mathbf{S}, \mathbf{T}, \mathbf{L}, \varepsilon)$-secure in the AI-RO+GG+IC$(\{M_i, N_i\}_{i=1}^{n_1}, \{p_i, \ell_i\}_{i=1}^{n_2}, \{K_i, C_i\}_{i=1}^{n_3})$*-model, for*

$$\varepsilon \leq 2\varepsilon' + 2\gamma,$$

where e is the Euler's number, and $(S_i)_G^{\mathsf{comb}}$, $(T_i)_G^{\mathsf{comb}}$ and $(L_{i,j})_G^{\mathsf{comb}}$ are the combined query complexity corresponding to G that corresponds to the random oracles, the generic group oracles, and the ideal ciphers, respectively.

C Multi-User Security of Key-Prefixed Short Schnorr Signatures

For completeness, we employ the bit-fixing-to-auxiliary-input technique utilized in Theorem 4.5 to examine the multi-user security of a *key-prefixed* Schnorr signature scheme. We then compare our findings with the prior work [6]. First, we analyze the multi-user security of key-prefixed Schnorr signatures within the bit-fixing ROM+GGM framework.

Theorem C.1. *Let* $\Pi^* = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ *be a* key-prefixed *Schnorr signature scheme and $p > 2^{2\lambda}$ be a prime number. Let $N \in \mathbb{N}$ be a parameter and $\left(\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \rightarrow \mathbb{G}$ such that $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ fixes P_1 input/output pairs of a random oracle $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^{\lambda_1}$ and P_2 input/output pairs of a generic group oracle $\tau : \mathbb{Z}_p \rightarrow \mathbb{G}$ such that $P_1 + P_2 = P$. If $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} \coloneqq q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles, at most $q_{\mathsf{H}}^{\mathsf{on}}$ queries to the random oracle, and at most $q_{\mathsf{S}}^{\mathsf{on}}$ queries to the signing oracle, then*

$$\Pr\left[\mathsf{SigForge}^{\tau,N}_{\mathcal{A}_{\mathsf{Sig.on,str}_{\tau,\mathsf{H}}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)},\Pi^*}(\lambda) = 1\right] \leq \varepsilon,$$

with

$$\varepsilon = \frac{q_{\mathsf{G}}^{\mathsf{on}}(N+P) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}}+1)/2}{p - 2P(N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1) - (N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 - N} + \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P)}{p} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P}{p - (N + P + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + 1}{2^{\lambda_1}},$$

where the randomness is taken over the selection of τ and the random coins of $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$.

Proof. Given a bit-fixing generic adversary $\left(\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ that attacks the short Schnorr signature scheme, we construct the following efficient generic algorithm $\mathcal{A}_{\mathsf{bridge}} = \left(\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{bridge.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ in the bit-fixing model which tries to succeed in the 1-out-of-N generic BRIDGE^N-finding game $\mathsf{BridgeChal}^{\tau,N}_{\mathcal{A}_{\mathsf{bridge}}}(\lambda)$:

Algorithm $\left(\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{bridge.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$:

The algorithm is given $p, \mathfrak{g} = \tau(1), \mathsf{pk}_i = \tau(x_i), 1 \leq i \leq N$ as input.

1. $\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ simply runs $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ and fixes the same P_1 input/output pairs of H $(\{(h_1, \mathsf{H}(h_1)), \ldots, (h_{P_1}, \mathsf{H}(h_{P_1}))\})$ and P_2 input/output pairs of τ $(\{(t_1, \tau(t_1)), \ldots, (t_{P_2}, \tau(t_{P_2}))\})$ as being fixed by $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$.
2. Initialize the set $\mathsf{H}_{\mathsf{resp}} = \{\}$ which stores the random oracle input/output pairs observed during online processing. It also maintains the set $\mathsf{H}_{\mathsf{Fixed}} = \{(h_1, \mathsf{H}(h_1)), \ldots, (h_{P_1}, \mathsf{H}(h_{P_1}))\}$ that $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ has already fixed in Step 1. Let $h_i = \mathsf{pk}_{j_i} \| I_i' \| m_i'$ for each $i \in [P_1]$ and for some $j_i \in [N]$.
3. Initialize the list $\mathcal{L} = \{(\tau(1), \mathbf{0}, 1), (\mathsf{pk}_i, \hat{u}_i, 0)$ for $i \in [N], (\tau(t_j), \mathbf{0}, t_j)$ for $j \in [P_2], (I_i', \mathbf{0}, r_i' \coloneqq \mathtt{DLog}(I_i'))$ for $i \in [P_1]\}$ that contains $1 + N + P$ tuples, and runs $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ with a number of access to the generic group oracles $\mathtt{GO} = (\mathtt{Mult}(\cdot,\cdot), \mathtt{Inv}(\cdot))$, $\mathtt{DLog}(\cdot)$, $\mathsf{Sign}_i(\cdot)$ for $1 \leq i \leq N$, and the random oracle $\mathsf{H}(\cdot)$. We maintain the invariant that every output of a generic group query during the online phase appears in the list $\mathcal{L}$. We consider the following cases:
   (a) Whenever $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ submits a query w to the random oracle H:
      - If there is a pair $(w, R) \in \mathsf{H}_{\mathsf{resp}}$ for some string $R \in \{0,1\}^{\lambda_1}$ then return R.
      - Otherwise, check if $(w, \mathsf{H}(w)) \in \mathsf{H}_{\mathsf{Fixed}}$. If so, then return $\mathsf{H}(w)$.
      - Otherwise, select $R \leftarrow\$ \{0,1\}^{\lambda_1}$ and add (w, R) to $\mathsf{H}_{\mathsf{resp}}$.
      - If w has the form $w = (\mathsf{pk}_j \| \mathfrak{a} \| m_i)$ where the value $\mathfrak{a}$ has not been observed previously (i.e., is not in the list $\mathcal{L}$ then we query $b = \mathtt{DLog}(\mathfrak{a})$ and add $(\mathfrak{a}, \mathbf{0}, b)$ to $\mathcal{L}$.

   (b) Whenever $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ submits a query $\mathfrak{a}$ to the generic group oracle $\mathtt{Inv}(\cdot)$:
      - If $\mathfrak{a}$ is not in $\mathcal{L}$ then we immediately query $b = \mathtt{DLog}(\mathfrak{a})$ and add $(\mathfrak{a}, \mathbf{0}, b)$ to $\mathcal{L}$.
      - Otherwise, $(\mathfrak{a}, \mathbf{a}, b) \in \mathcal{L}$ for some $\mathbf{a}$ and b. Then we query $\mathtt{Inv}(\mathfrak{a}) = \tau(-\mathbf{a} \cdot \mathbf{x} - b)$, output the result and add $(\tau(-\mathbf{a} \cdot \mathbf{x} - b), -\mathbf{a}, -b)$ to $\mathcal{L}$.

   (c) Whenever $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ submits a query $\mathfrak{a}, \mathfrak{b}$ to the generic group oracle $\mathtt{Mult}(\cdot,\cdot)$:
      - If the element $\mathfrak{a}$ (resp. $\mathfrak{b}$) is not in $\mathcal{L}$ then query $b_0 = \mathtt{DLog}(\mathfrak{a})$ (resp. $b_1 = \mathtt{DLog}(\mathfrak{b})$) and add the element $(\mathfrak{a}, \mathbf{0}, b_0)$ (resp. $(\mathfrak{b}, \mathbf{0}, b_1)$) to $\mathcal{L}$.
      - Otherwise both elements $(\mathfrak{a}, \mathbf{a}_0, b_0), (\mathfrak{b}, \mathbf{a}_1, b_1) \in \mathcal{L}$. Then we return $\mathtt{Mult}(\mathfrak{a}, \mathfrak{b}) = \tau((\mathbf{a}_0 + \mathbf{a}_1) \cdot \mathbf{x} + b_0 + b_1)$ and add $(\tau((\mathbf{a}_0 + \mathbf{a}_1) \cdot \mathbf{x} + b_0 + b_1), \mathbf{a}_0 + \mathbf{a}_1, b_0 + b_1) \in \mathcal{L}$.

   (d) Whenever $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ submits a query m_i to the signing oracle $\mathsf{Sign}(x_j, \cdot)$:

- The attacker tries to forge a signature without knowledge of the secret key x_j, relying on the ability to program the random oracle as follows:
  (i) Pick s_i, e_i randomly and compute $I_i = \tau(s_i - x_j e_i) = \texttt{Mult}(\mathfrak{s}_i, \mathsf{Pow}(\mathsf{pk}_j, -e_i))$ where $\mathfrak{s}_i = \mathsf{Pow}(\mathfrak{g}, s_i)$.
  (ii) If $\mathsf{H}(\mathsf{pk}_j \| I_i \| m_i)$ has been previously queried or if it is in the set $\mathsf{H}_{\mathsf{Fixed}}$, then return $\perp$.
  (iii) Otherwise, program $\mathsf{H}(\mathsf{pk}_j \| I_i \| m_i) := e_i$ and return $\sigma_i = (s_i, e_i)$.
- We remark that a side effect of querying the Sign_j oracle is the addition of the tuples $(\tau(s_i), \mathbf{0}, s_i)$, $(\tau(x_j e_i), e_i \hat{u}_i, 0)$ and $(\tau(s_i - x_j e_i), -e_i \hat{u}_i, s_i)$ to $\mathcal{L}$, since these values are computed using the generic group oracles $\texttt{Inv}, \texttt{Mult}$.

4. After $\mathcal{A}_{\mathsf{sig}}^{\mathsf{on}}$ outputs $\sigma_{i*} = (s_{i*}, e_{i*})$ and m_{i*}, identify the index $i* \in [N]$ such that $\mathsf{Vf}(\mathsf{pk}_{i*}, m_{i*}, \sigma_{i*}) = 1$. Without loss of generality, we can assume that m_{i*} is not a part of pre-fixed points $\mathsf{H}_{\mathsf{Fixed}}$ in Step 2 since if the attacker forges a signature (s_{i*}, e_{i*}) for a message m_{i*} which involves the pre-fixed point $(\mathsf{pk}_{i*} \| I_{i*} \| m_{i*}, \mathsf{H}(\mathsf{pk}_{i*} \| I_{i*} \| m_{i*})) \in \mathsf{H}_{\mathsf{Fixed}}$ then we can directly force the bridge event to occur, i.e., $(I_{i*}, \mathbf{0}, r_{i*} = \mathsf{DLog}(I_{i*})) \in \mathcal{L}$ and $(I_{i*}, -e_{i*}\hat{u}_{i*}, s_{i*}) \in \mathcal{L}$ (see Step 7 below) since $\mathbf{0} \neq -e_{i*}\hat{u}_{i*}$.
5. Compute $\tau(-e_{i*}x_{i*}) = \texttt{Inv}(\mathsf{Pow}(\tau(x_{i*}), e_{i*}))$. This will ensure that the elements $(\tau(-e_{i*}x_{i*}), -e_{i*}\hat{u}_{i*}, 0)$ and $(\tau(e_{i*}x_{i*}), e_{i*}\hat{u}_{i*}, 0)$ are both added to $\mathcal{L}$.
6. Compute $\mathfrak{s}_{i*} = \mathsf{Pow}(\mathfrak{g}, s_{i*})$ to ensure that $(\mathfrak{s}_{i*}, \mathbf{0}, s_{i*}) \in \mathcal{L}$.
7. Finally, compute $I_{i*} = \texttt{Mult}(\mathfrak{s}_{i*}, \tau(-e_{i*}x_{i*})) = \tau(s_{i*} - x_{i*}e_{i*})$ which ensures that $(I_{i*}, -e_{i*}\hat{u}_{i*}, s_{i*}) \in \mathcal{L}$ and check to see if we previously had any tuple of the form $(I_{i*}, \mathbf{a}, b) \in \mathcal{L}$.

Analysis. We first remark that if the signature is valid then we must have $e_{i*} = \mathsf{H}(\mathsf{pk}_{i*} \| I_{i*} \| m_{i*})$ and $\mathsf{DLog}(I_{i*}) = s_{i*} - x_{i*}e_{i*} = \mathbf{a} \cdot \mathbf{x} + b$. Moreover, without loss of generality, we can assume that each string $\mathfrak{y}$ occurs at most once in the list $\mathcal{L}$ in Step 3 because if at any point we have some string $\mathfrak{y}$ such that $(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L}$ and $(\mathfrak{y}, \mathbf{c}, d) \in \mathcal{L}$ for $(\mathbf{a}, b) \neq (\mathbf{c}, d)$ then we can immediately have a BRIDGE^N instance $(\tau((\mathbf{a} - \mathbf{c}) \cdot \mathbf{x}), \mathbf{a} - \mathbf{c}, 0) \in \mathcal{L}$ and $(\tau(d - b), \mathbf{0}, d - b) \in \mathcal{L}$ since $\tau((\mathbf{a} - \mathbf{c}) \cdot \mathbf{x}) = \tau(d - b)$.

We now consider the failure events that our algorithm outputs $\perp$ for failure, either before $\mathcal{A}_{\mathsf{Sig}}^{\mathsf{on}}$ outputs a signature or after $\mathcal{A}_{\mathsf{Sig}}^{\mathsf{on}}$ outputs a valid signature.

(1) The first failure event we consider is when our algorithm outputs $\perp$ before $\mathcal{A}_{\mathsf{Sig}}^{\mathsf{on}}$ outputs a signature. This event is called $\mathsf{FailtoSign}$ when our reduction outputs $\perp$ in Step 3.(d) due to the signing oracle failure, i.e., since $\mathsf{H}(\mathsf{pk}_j \| I_i \| m_i)$ has been previously queried or it is contained in the set $\mathsf{H}_{\mathsf{Fixed}}$. We observe that every time the attacker queries the signing oracle, we would generate a query to the random oracle H. Thus, we would have at most $q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1$ input/output pairs recorded for the random oracle, including the fixed points during the preprocessing phase. Since $I_i = \tau(s_i - x_i e_i)$ represents a fresh/randomly selected group element of size p, the probability that $(\mathsf{pk}_j, I_i, m_i)$ is one of the inputs is at most $(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)/p$. Applying union bound over $q_{\mathsf{S}}^{\mathsf{on}}$ queries to the signing oracle, we have that

$$\Pr[\mathsf{FailtoSign}] \leq \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)}{p}.$$

(2) Our algorithm can also output $\perp$ even after $\mathcal{A}_{\mathsf{Sig}}^{\mathsf{on}}$ outputs a valid signature. We define the event $\mathsf{FailtoFind}(I_{i*})$ that we find that the signature is valid but I_{i*} was not previously recorded in our list $\mathcal{L}$ before we computed $\texttt{Mult}(\mathfrak{s}_{i*}, \tau(-e_{i*}x_{i*}))$ in the last step so that we cannot find the bridge event. We observe that if $I_{i*} \notin \mathcal{L}$ then we can view $I_{i*} = \tau(s_{i*} - x_{i*}e_{i*})$ as a uniformly random binary string from a set of size at least $p - |\mathcal{L}|$ which had not yet been selected at the time $\mathcal{A}_{\mathsf{Sig}}^{\mathsf{on}}$ outputs σ_{i*}. Thus, the probability that the query $\mathsf{H}(\mathsf{pk}_{i*} \| I_{i*} \| m_{i*})$ was previously recorded is at most $(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)/(p - |\mathcal{L}|)$. If the query $\mathsf{H}(\mathsf{pk}_{i*} \| I_{i*} \| m_{i*})$ was *not* previously recorded then the probability of a successful forgery $\mathsf{H}(\mathsf{pk}_{i*} \| I_{i*} \| m_{i*}) = e_{i*}$ is *at most* 2^{-k_1} since we can view $\mathsf{H}(\mathsf{pk}_{i*} \| I_{i*} \| m_{i*})$ chosen uniformly at random from the possible 2^{λ_1} options. Hence, we have that

$$\Pr[\mathsf{FailtoFind}(I_{i*})] \leq \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1}{p - |\mathcal{L}|} + \frac{1}{2^{\lambda_1}}.$$

(3) Finally, there is another failure event called $\mathsf{BadQuery}$ after $\mathcal{A}_{\mathsf{Sig}}^{\mathsf{on}}$ outputs a valid signature, which denotes the event that the signature is valid but for the only prior tuple $(I_{i*}, \mathbf{a}, b) \in \mathcal{L}$ we have that $\mathbf{a} = -e_{i*}\hat{u}_{i*}$ so that we still cannot find the bridge event. We observe that by construction we ensure that the tuple $(I, \mathbf{a}, b)$ will always be recorded in $\mathcal{L}$ before a query of the form $\mathsf{H}(\mathsf{pk}_i \| I \| m)$ is ever issued — if I is new then we call $\mathsf{DLog}(I)$ before querying the random oracle. Now define a subset $\widehat{\mathcal{L}} \subset \mathcal{L}$ as the set of tuples $(\mathfrak{a}, \mathbf{a}, b) \in \mathbb{G} \times \mathbb{Z}_p^N \times \mathbb{Z}_p$ such that $\mathbf{a}$ has exactly *one* nonzero element. Now we call a random oracle query $x = (\mathsf{pk}_i \| I \| m)$ "bad" if $\mathsf{H}(x) = -\widetilde{a}$ where the tuple $(I, \mathbf{a}, b) \in \widehat{\mathcal{L}}$ has already been recorded and the nonzero element of $\mathbf{a}$ is $\widetilde{a}$ (Recall that if there were two recorded tuples $(I, \mathbf{a}, b)$ and $(I, \mathbf{c}, d)$ then our algorithm would have already found a BRIDGE^N instance). Thus, the probability each individual query is "bad" is at most $1/2^{\lambda_1}$ and we can use union bounds to upper bound the probability of any "bad" query as

$$\Pr[\mathsf{BadQuery}] \leq \frac{q_{\mathsf{H}}^{\mathsf{on}}}{2^{\lambda_1}}.$$

Now we have shown that

$$\begin{aligned}
&\Pr\left[\mathsf{BridgeChal}_{\mathcal{A}_{\mathsf{bridge}}}^{\tau,N}(\lambda) = 1\right] \\
&\geq \Pr\left[\mathsf{SigForge}_{\mathcal{A}_{\mathsf{Sig.on,str}_{\tau,\mathsf{H}}}^{\mathsf{BF\text{-}RO}}, \Pi_{\setminus\{0\}}^{\mathsf{short}}}^{\tau,N}(\lambda) = 1\right] - \Pr[\mathsf{FailtoSign}] \\
&\quad - \Pr[\mathsf{FailtoFind}(I_{i*})] - \Pr[\mathsf{BadQuery}]
\end{aligned}$$

$$\geq \Pr\left[\mathsf{SigForge}^{\tau,N}_{\mathcal{A}^{\mathsf{BF\text{-}RO}}_{\mathsf{Sig.on},\mathsf{str}_{\tau,\mathsf{H}}},\Pi^{\mathsf{short}}_{\setminus\{0\}}}(\lambda)=1\right] - \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+P_1)}{p} - \frac{q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+P_1}{p-|\mathcal{L}|} - \frac{q_{\mathsf{H}}^{\mathsf{on}}+1}{2^{\lambda_1}}.$$

Finally, by applying Lemma 4.3, we can conclude that

$$\Pr\left[\mathsf{SigForge}^{\tau,N}_{\mathcal{A}^{\mathsf{BF\text{-}RO}}_{\mathsf{Sig.on},\mathsf{str}_{\tau,\mathsf{H}}},\Pi^{\mathsf{short}}_{\setminus\{0\}}}(\lambda)=1\right]$$
$$\leq \left[\mathsf{BridgeChal}^{\tau,N}_{\mathcal{A}_{\mathsf{bridge}}}(\lambda)=1\right] + \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+P_1)}{p} + \frac{q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+P_1}{p-|\mathcal{L}|} + \frac{q_{\mathsf{H}}^{\mathsf{on}}+1}{2^{\lambda_1}}$$
$$\leq \frac{q_{\mathsf{G}}^{\mathsf{on}}(N+P)+3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}}+1)/2}{p-2P(N+3q_{\mathsf{G}}^{\mathsf{on}}+1)-(N+3q_{\mathsf{G}}^{\mathsf{on}}+1)^2-N} + \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+P)}{p} + \frac{q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+P}{p-(N+P+3q_{\mathsf{G}}^{\mathsf{on}}+1)} + \frac{q_{\mathsf{H}}^{\mathsf{on}}+1}{2^{\lambda_1}}. \quad \square$$

Combining with Theorem 3.2, we get the following result in the AI-ROM+GGM.

THEOREM C.2. *Let* $\Pi^* = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ *be a* key-prefixed *Schnorr signature scheme,* $p > 2^{2\lambda}$ *be a prime number, and* $\gamma > 0$ *be a parameter. Let* $N \in \mathbb{N}$ *be a parameter and* $\left(\mathcal{A}^{\mathsf{AI\text{-}RO+GG}}_{\mathsf{Sig.pre}}, \mathcal{A}^{\mathsf{AI\text{-}RO+GG}}_{\mathsf{Sig.on}}\right)$ *be a pair of generic algorithms with a labeling map* $\tau : \mathbb{Z}_p \longrightarrow \mathbb{G}$ *such that* $\mathcal{A}^{\mathsf{AI\text{-}RO+GG}}_{\mathsf{Sig.pre}}$ *outputs an S-bit hint* $\mathsf{str}_{\mathsf{H},\tau}$. *If* $\mathcal{A}^{\mathsf{AI\text{-}RO+GG}}_{\mathsf{Sig.on}}$ *takes* $\mathsf{str}_{\mathsf{H},\tau}$ *as input and makes at most* $q_{\mathsf{G}}^{\mathsf{on}} \coloneqq q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ *queries to the generic group oracles, at most* $q_{\mathsf{H}}^{\mathsf{on}}$ *queries to the random oracle, and at most* $q_{\mathsf{S}}^{\mathsf{on}}$ *queries to the signing oracle, then* $\Pr\left[\mathsf{SigForge}^{\tau,N}_{\mathcal{A}^{\mathsf{AI\text{-}RO+GG}}_{\mathsf{Sig.on},\mathsf{str}_{\tau,\mathsf{H}}},\Pi^*}(\lambda)=1\right] \leq \varepsilon$*, with*

$$\varepsilon = \frac{2q_{\mathsf{G}}^{\mathsf{on}}(N+Sq)+3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}}+1)}{p-2Sq(N+3q_{\mathsf{G}}^{\mathsf{on}}+1)-(N+3q_{\mathsf{G}}^{\mathsf{on}}+1)^2-N} + \frac{2q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+Sq)}{p} + \frac{2(q_{\mathsf{H}}^{\mathsf{on}}+q_{\mathsf{S}}^{\mathsf{on}}+Sq)}{p-(N+Sq+3q_{\mathsf{G}}^{\mathsf{on}}+1)} + \frac{q_{\mathsf{H}}^{\mathsf{on}}+1}{2^{\lambda_1-1}} + 2^{-2\lambda+1},$$

where $q = q_{\mathsf{H}}^{\mathsf{on}} + 3q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}}$ *and the randomness is taken over the selection of* τ *and the random coins of* $\mathcal{A}^{\mathsf{AI\text{-}RO+GG}}_{\mathsf{Sig.on}}$.

PROOF. This is straightforward by combining Theorem C.1 with Theorem 3.1 and by setting an optimal $P \approx Sq$ and $\gamma = 2^{-2\lambda}$. $\square$

D Warmup: CPA Security of PSEC-KEM without Preprocessing in the ROM+GGM

The Generic Quadratic Bridge-Finding Game. We establish the CPA security of PSEC-KEM (without preprocessing) via reduction from a new game called *the generic quadratic bridge-finding game.* As the name suggests, we will consider the game in the GGM with the labeling map $\tau : \mathbb{Z}_p \longrightarrow \mathbb{G}$. In this game, the attacker is given $\tau(1)$ and $\tau(x)$ for a random unknown $x \in \mathbb{Z}_p$. The attacker is given access to the generic group oracle $(\mathtt{Mult}, \mathtt{Inv})$, along with an additional oracle $\mathcal{O}$ which returns some $\tau(\alpha)$ where the value $\alpha \in \mathbb{Z}_p$ is unknown to the attacker. In a nutshell, as the attacker makes multiple queries to those oracles, the number of unknowns can be increased by at most 2 per query, and the attacker's goal is to find a specific form of quadratic relationship between these unknowns. One important note is that the attacker *must output* the very quadratic relationship. It is not required for the attacker to solve the equation for the unknowns, but the attacker should specify which unknown values satisfy which quadratic relationship. We formally define the game below.

The Generic Quadratic Bridge-Finding Game
$\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda)$:

1. The challenger picks a random $x \leftarrow\!\$\, \mathbb{Z}_p$ and initializes the list $\mathcal{L} = \{(\tau(1), 0, 1), (\tau(x), 1, 0)\}$.
2. The adversary $\mathcal{A}$ is given $\tau(1)$ and $\tau(x)$ as input. Here, $\tau : \mathbb{Z}_p \longrightarrow \mathbb{G}$ is a labeling map where $\mathbb{G}$ is a set of bitstrings of length $\ell \geq \log p$, and p is a 2λ-bit prime.
3. The challenger maintains a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ which is a vector of unknowns such that $x_1 = x$, where the dimension n satisfies the following:
   - initially, $n = 1$ since the only unknown is x, and
   - the number of unknowns n increases by at most 2 as $\mathcal{A}$ makes query to the generic group oracle $(\mathtt{Mult}, \mathtt{Inv})$ or a generic oracle $\mathcal{O}$ that selects a new random unknown variable $x_j \leftarrow\!\$\, \mathbb{Z}_p$ and outputs $\tau(x_j)$..
4. $\mathcal{A}$ is allowed to query the generic group oracle $(\mathtt{Mult}(\cdot,\cdot), \mathtt{Inv}(\cdot))$ and the oracle $\mathcal{O}$.
   (a) If $\mathcal{A}$ submits a query to $\mathcal{O}$, $\mathcal{A}$ gets some bitstring $\mathfrak{y} \in \tau(\mathbb{Z}_p)$. If $\mathfrak{y}$ does not appear in $\mathcal{L}$ and $\dim(\mathbf{x}) = j$, then the challenger updates $\mathbf{x} \longleftarrow \mathbf{x} \circ x_{j+1}$ (i.e., increase the dimension of $\mathbf{x}$ by 1 and add the element x_{j+1} at the end), where x_{j+1} denotes an unknown variable satisfying the constraint $\tau(x_{j+1}) = \mathfrak{y}$, and updates all of the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$. Then the challenger adds $(\mathfrak{y}, \mathbf{0} \circ 1, 0)$ to $\mathcal{L}$.
   (b) If $\mathcal{A}$ submits a query $\mathfrak{y}_1, \mathfrak{y}_2$ to $\mathtt{Mult}(\cdot,\cdot)$:
      - If there exist tuples $(\mathfrak{y}_1, \mathbf{a}_1, b_1), (\mathfrak{y}_2, \mathbf{a}_2, b_2) \in \mathcal{L}$ for both $\mathfrak{y}_1$ and $\mathfrak{y}_2$, then the challenger adds the tuple $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{a}_1 + \mathbf{a}_2, b_1 + b_2)$ to the list $\mathcal{L}$.
      - If $(\mathfrak{y}_1, \mathbf{a}_1, b_1) \in \mathcal{L}$ but $\mathfrak{y}_2$ does not appear in $\mathcal{L}$ and if $\mathfrak{y}_2 \in \tau(\mathbb{Z}_p)$, and if $\dim(\mathbf{x}) = j$, then the challenger first updates $\mathbf{x} \longleftarrow \mathbf{x} \circ x_{j+1}$ where x_{j+1} denotes an unknown variable satisfying the constraint $\tau(x_{j+1}) = \mathfrak{y}_2$ and updates all of the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$. Then the challenger adds tuples $(\mathfrak{y}_2, \mathbf{0} \circ 1, 0)$ and $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{a}_1 \circ 0 + \mathbf{0} \circ 1, b_1)$ to $\mathcal{L}$. Do it similarly if $\mathfrak{y}_2$ appears in $\mathcal{L}$ and $\mathfrak{y}_1$ does not.
      - If both $\mathfrak{y}_1$ and $\mathfrak{y}_2$ does not appear in $\mathcal{L}$, and if both $\mathfrak{y}_1, \mathfrak{y}_2 \in \tau(\mathbb{Z}_p)$, and if $\dim(\mathbf{x}) = j$, then the challenger first updates $\mathbf{x} \longleftarrow \mathbf{x} \circ x_{j+1} \circ x_{j+2}$

where x_{j+1} and x_{j+2} are unknown values such that $\tau(x_{j+i}) = \mathfrak{y}_i$ for $i = 1, 2$, and updates all the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0^2, b)$. Then the challenger adds tuples $(\mathfrak{y}_1, \mathbf{0} \circ 1 \circ 0, 0)$, $(\mathfrak{y}_2, \mathbf{0} \circ 0 \circ 1, 0)$, and $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{0} \circ 1 \circ 1, 0)$ to $\mathcal{L}$.

(c) If $\mathcal{A}$ submits a query $\mathfrak{y}$ to $\mathtt{Inv}(\cdot)$:
- If there exists a tuple $(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L}$, then the challenger adds the tuple $(\mathtt{Inv}(\mathfrak{y}), -\mathbf{a}, -b)$ to the list $\mathcal{L}$.
- If $\mathfrak{y}$ does not appear in $\mathcal{L}$ and $\mathfrak{y} \in \tau(\mathbb{Z}_p)$, and if $\dim(\mathbf{x}) = j$, then the challenger first updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1}$ where x_{j+1} is an unknown value that satisfies $\tau(x_{j+1}) = \mathfrak{y}$ and updates all of the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$. Then the challenger adds tuples $(\mathfrak{y}, \mathbf{0} \circ 1, 0)$ and $(\mathtt{Inv}(\mathfrak{y}), \mathbf{0} \circ (-1), 0)$ to $\mathcal{L}$.

5. If at any point in time $\mathcal{A}$ outputs a tuple $(i_1, i_2, \mathbf{a}, b)$ such that (1) $1 \le i_1, i_2 \le \dim(\mathbf{x})$, (2) $(\mathbf{a}, b) \in \mathbb{Z}_p^{\dim(\mathbf{x})} \times \mathbb{Z}_p$, and (3) $x_{i_1} x_{i_2} = \mathbf{a} \cdot \mathbf{x} + b \mod p$, then we say that the event QDBridge occurs. In this case, the output of the game is 1 and we write $\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1$. If QDBridge never occurs, then the output of the game is 0.

Lemma D.1 upper bounds the probability to win the generic quadratic bridge-finding game $\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda)$.

Lemma D.1. *For any attacker $\mathcal{A}$ making at most q_{G} generic group oracle queries and q_O queries to the oracle O,*

$$\Pr\left[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1\right] \le \frac{3(q_{\mathsf{G}}+q_O)^2+5(q_{\mathsf{G}}+q_O)+4}{2p-2(3q_{\mathsf{G}}+q_O+1)^2-2(2q_{\mathsf{G}}+q_O)},$$

in the generic group model of prime order p where the randomness is taken over the selection of x and τ, as well as any random coins of $\mathcal{A}$.

Proof. The proof works by maintaining a list $\mathcal{L}$ of tuples $(\tau(y), \mathbf{a}, b)$ where $\tau(y) \in \mathbb{G}$, $\mathbf{a} \in \mathbb{Z}_p^{\dim(\mathbf{x})}$, and $b \in \mathbb{Z}_p$ such that $y = \mathbf{a} \cdot \mathbf{x} + b$ for every oracle output $\tau(y)$. To bound the probability that the attacker wins the quadratic bridge-finding game, we consider the case where the event QDBridge occurs after the attacker makes q_{G} generic group queries and q_O queries to O. Suppose that $\dim(\mathbf{x}) = j$ at the point in time when QDBridge occurs. We observe that $j \le 2q_{\mathsf{G}} + q_O + 1$ since $\dim(\mathbf{x})$ can be increased by at most 2 whenever $\mathcal{A}$ makes a generic group query and at most 1 whenever $\mathcal{A}$ makes a query to O. We further observe that in this case, $|\mathcal{L}| \le 3q_{\mathsf{G}} + q_O + 1$ since for each query to O, at most 1 item can be added to $\mathcal{L}$ whereas each generic group oracle query can add at most 3 itmes to $\mathcal{L}$ (exactly three in the case when $\mathcal{A}$ queries $\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2)$ on two fresh elements).

Now consider the event BRIDGE where the list $\mathcal{L}$ contains two distinct tuples $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$ such that $\mathbf{a} \cdot \mathbf{x} + b = \mathbf{a}' \cdot \mathbf{x} + b'$. To upper bound Pr[BRIDGE], consider the event $\overline{\mathsf{BRIDGE}_{<i}}$ that the event BRIDGE has not occurred until the (i − 1)th query. Conditioning on the event $\overline{\mathsf{BRIDGE}_{<i}}$, we are now interested in the event BRIDGE_i where the ith query makes the event BRIDGE occur, i.e., a tuple $(\mathfrak{y}_i, \mathbf{a}_i, b_i)$ has been recorded to $\mathcal{L}$ and there exists a tuple of the form $(\cdot, \mathbf{a}, b)$ such that $\mathbf{a}_i \cdot \mathbf{x} + b_i = \mathbf{a} \cdot \mathbf{x} + b$. We can essentially view $\mathbf{x}$ sampled uniformly at random subject to some restrictions due to $\overline{\mathsf{BRIDGE}_{<i}}$, from which we know that for any distinct pair $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$ we have $\mathbf{a} \cdot \mathbf{x} + b \ne \mathbf{a}' \cdot \mathbf{x} + b'$.

Let $r \le \dim(\mathbf{x}) = j$ be an index such that $\mathbf{a}[r] - \mathbf{a}'[r] \ne 0$ and suppose that $x_r = \mathbf{x}[r]$ is the last value sampled. At this point, we can view x_r as being drawn uniformly at random from a set of at least $p - |\mathcal{L}|^2 - (j-1)$ remaining values. To see this, we observe that x_r must be distinct from $\{x_1, \ldots, x_j\} \setminus \{x_r\}$ since otherwise we would have had the event $\mathsf{BRIDGE}_{<i}$, and for each pair of distinct elements $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$, we can eliminate one possible value of x_r due to the fact that $\mathbf{a} \cdot \mathbf{x} + b \ne \mathbf{a}' \cdot \mathbf{x} + b'$ and $\mathbf{a}[r] \ne \mathbf{a}'[r]$, which implies $x_r \ne \left[(b' - b) + \sum_{i \ne r} x_i(\mathbf{a}'[i] - \mathbf{a}[i])\right] \cdot (\mathbf{a}[r] - \mathbf{a}'[r])^{-1}$. Thus, the probability that $\mathbf{a}_i \cdot \mathbf{x} + b_i = \mathbf{a} \cdot \mathbf{x} + b$ is at most $\frac{1}{p-(3q_{\mathsf{G}}+q_O+1)^2-(2q_{\mathsf{G}}+2q_O)}$. Union bounding over all tuples in $\mathcal{L}$, we have

$$\begin{aligned}\Pr[\mathsf{BRIDGE}] &= \sum_{i \le q_{\mathsf{G}}+q_O} \Pr\left[\mathsf{BRIDGE}_i \middle| \overline{\mathsf{BRIDGE}_{<i}}\right] \\ &\le \sum_{i \le q_{\mathsf{G}}+q_O} \frac{1+3i}{p - (3q_{\mathsf{G}} + q_O + 1)^2 - (2q_{\mathsf{G}} + 2q_O)} \\ &\le \frac{3(q_{\mathsf{G}} + q_O)^2 + 5(q_{\mathsf{G}} + q_O)}{2p - 2(3q_{\mathsf{G}} + q_O + 1)^2 - 2(2q_{\mathsf{G}} + 2q_O)}.\end{aligned}$$

Now, consider the case where the event BRIDGE never occurs. Then we would like to upper bound the probability

$$\Pr\left[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1 \middle| \overline{\mathsf{BRIDGE}}\right].$$

The argument works similarly; conditioning on the event $\overline{\mathsf{BRIDGE}}$, we know that for any distinct pair $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$ we have $\mathbf{a} \cdot \mathbf{x} + b \ne \mathbf{a}' \cdot \mathbf{x} + b'$. Let $r \le \dim(\mathbf{x}) \le 2q_{\mathsf{G}} + q_O + 1$ be an index such that $\mathbf{a}[r] - \mathbf{a}'[r] \ne 0$ and suppose that $x_r = \mathbf{x}[r]$ is the last value sampled. At this point, we can view x_r as being drawn uniformly at random from a set of at least $p - |\mathcal{L}|^2 - (2q_{\mathsf{G}} + q_O)$ remaining values as before. Now, the attacker wins if and only if s/he outputs a tuple $(i_1, i_2, \mathbf{a}, b)$ such that $x_{i_1} x_{i_2} = \mathbf{a} \cdot \mathbf{x} + b$. Since this equation is at most degree 2 in terms of x_r (in case $i_1 = i_2 = r$), there are at most 2 roots in $\mathbb{Z}_p$. Thus, we have that

$$\begin{aligned}&\Pr\left[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1 \middle| \overline{\mathsf{BRIDGE}}\right] \\ &\le \frac{2}{p - |\mathcal{L}|^2 - (2q_{\mathsf{G}} + q_O)} \\ &\le \frac{2}{p - (3q_{\mathsf{G}} + q_O + 1)^2 - (2q_{\mathsf{G}} + q_O)}.\end{aligned}$$

Taken together, we have

$$\begin{aligned}&\Pr\left[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1\right] \\ &\le \Pr[\mathsf{BRIDGE}] + \Pr\left[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1 \middle| \overline{\mathsf{BRIDGE}}\right] \\ &\le \frac{(3/2)(q_{\mathsf{G}} + q_O)^2 + (5/2)(q_{\mathsf{G}} + q_O)}{p - (3q_{\mathsf{G}} + q_O + 1)^2 - (2q_{\mathsf{G}} + q_O)} \\ &\quad + \frac{2}{p - (3q_{\mathsf{G}} + q_O + 1)^2 - (2q_{\mathsf{G}} + q_O)} \\ &= \frac{3(q_{\mathsf{G}} + q_O)^2 + 5(q_{\mathsf{G}} + q_O) + 4}{2p - 2(3q_{\mathsf{G}} + q_O + 1)^2 - 2(2q_{\mathsf{G}} + q_O)}.\end{aligned}$$

□

Security Reduction. Now we are ready to prove the CPA security of PSEC-KEM in the ROM+GGM by reduction from the quadratic bridge-finding game.

Theorem D.2. *The PSEC-KEM scheme* $\Pi = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* $(q_{\mathsf{H}}, q_{\mathsf{G}}, q_{\mathsf{E}}, \varepsilon)$*-CPA secure with*

$$\varepsilon = \frac{3(q_{\mathsf{G}} + 2q_{\mathsf{E}})^2 + 5(q_{\mathsf{G}} + 2q_{\mathsf{E}}) + 4}{p - (3q_{\mathsf{G}} + 2q_{\mathsf{E}} + 1)^2 - 2(q_{\mathsf{G}} + q_{\mathsf{E}})} + \frac{3q_{\mathsf{H}}q_{\mathsf{E}}}{2^{\lambda_1}} + \frac{q_{\mathsf{H}}q_{\mathsf{E}}}{p},$$

in the generic group model of order $p \approx 2^{2\lambda}$ *and the programmable random oracle model.*

Proof. To prove Theorem D.2, we use a hybrid argument to prove the CPA security of PSEC-KEM. In the first hybrid (hybrid H_0), the distinguisher $\mathcal{D}$ is given the CPA indistinguishability game for PSEC-KEM with the challenge bit $b = 0$, and in the last hybrid (hybrid H_3), $\mathcal{D}$ is given the CPA indistinguishability game with $b = 1$. Each hybrid is defined as follows:

Hybrid H_0. This is the original CPA indistinguishability game $\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda)$ for PSEC-KEM with the challenge bit $b = 0$. In particular, $\mathcal{D}$ has access to the encapsulation oracle $\mathsf{Encaps}'_0(\cdot) \coloneqq \mathsf{Encaps}(\cdot, 1^\lambda)$, i.e., whenever $\mathcal{D}$ submits a query $\mathsf{Encaps}'_0(\mathsf{pk})$, $\mathcal{D}$ always gets the output $\mathsf{Encaps}(\mathsf{pk}, 1^\lambda)$.

Hybrid H_1. This is the same security game with H_0 except that the encapsulation oracle $\mathsf{Encaps}'_0(\cdot)$ is replaced with a modified oracle $\mathsf{Encaps}'_1(\cdot)$. In $\mathsf{Encaps}'_1(\cdot)$, $\tau(\alpha)$ and $\tau(\alpha x)$ (where x is the secret key) are replaced with random elements in $\tau(\mathbb{Z}_p)$ by querying the oracle $\mathcal{O}$ twice. See Figure 2 for the details.

Hybrid H_2. This is the same security game with H_1 except that the encapsulation oracle $\mathsf{Encaps}'_1(\cdot)$ is further replaced with a modified oracle $\mathsf{Encaps}'_2(\cdot)$. $\mathsf{Encaps}'_2(\cdot)$ is exactly the same as $\mathsf{Encaps}'_1(\cdot)$ except that the key k is sampled uniformly at random from $\{0,1\}^\lambda$.

Hybrid H_3. This is the same security game with H_2 except that the encapsulation oracle $\mathsf{Encaps}'_2(\cdot)$ is replaced with a modified oracle $\mathsf{Encaps}'_3(\cdot) \coloneqq \mathsf{Encaps}_1(\cdot)$, which leads to the original CPA indistinguishability game $\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda)$ for PSEC-KEM with the challenge bit $b = 1$. In particular, $\mathsf{Encaps}'_3(\cdot)$ is the same as $\mathsf{Encaps}'_2(\cdot)$ but the random elements in $\tau(\mathbb{Z}_p)$ are reverted back to the honest computation $\tau(\alpha)$ and $\tau(\alpha x)$. This can also be interpreted as replacing an honest key k in $\mathsf{Encaps}'_0(\cdot)$ with a uniformly random key of length λ. See Figure 2 for the details.

Indistinguishability of H_0 *and* H_1. We observe that two hybrids are perfectly indistinguishable unless one of the two events occurs: for some round $i \le q_{\mathsf{E}}$,

(1) the random oracle query $\mathsf{H}_2(\tau(\alpha_i)\|\tau(\alpha_i x))$ has been made from the adversary, or
(2) the adversary makes query r_i to H_0 but has not queried $(\tau(\alpha_i)\|\tau(\alpha_i x))$ to H_2.

In Case (1), we argue that $\mathcal{D}$ can win the quadratic bridge-finding game. Once $\mathcal{D}$ ever makes a query of the form $\mathsf{H}_2(\tau(\alpha_i)\|\tau(\alpha_i x))$, $\mathcal{D}$ can win the quadratic bridge-finding game as follows:

- We observe that $\mathcal{D}$ can maintain the list $\mathcal{L}$ as follows.

```
Encaps'_0(pk)
1 :  $r \hookleftarrow\$ \{0,1\}^{\lambda_1}$
2 :  $\alpha \leftarrow \mathsf{H}_0(r),\ k \leftarrow \mathsf{H}_1(r)$
3 :  $c_1 \leftarrow r \oplus \mathsf{H}_2(\tau(\alpha)\|\tau(\alpha x))$
4 :  $c \leftarrow (\tau(\alpha), c_1)$
5 :  Return $(k, c)$
```

```
Encaps'_1(pk)
1 :  $r \hookleftarrow\$ \{0,1\}^{\lambda_1}$
2 :  $\mathfrak{y}_1 \leftarrow \mathcal{O}(\cdot), \mathfrak{y}_2 \leftarrow \mathcal{O}(\cdot),\ k \leftarrow \mathsf{H}_1(r)$
3 :  $c_1 \leftarrow r \oplus \mathsf{H}_2(\mathfrak{y}_1\|\mathfrak{y}_2)$
4 :  $c \leftarrow (\mathfrak{y}_1, c_1)$
5 :  Return $(k, c)$
```

```
Encaps'_2(pk)
1 :  $r \hookleftarrow\$ \{0,1\}^{\lambda_1}$
2 :  $\mathfrak{y}_1 \leftarrow \mathcal{O}(\cdot), \mathfrak{y}_2 \leftarrow \mathcal{O}(\cdot), k \hookleftarrow\$ \{0,1\}^{\lambda}$
3 :  $c_1 \leftarrow r \oplus \mathsf{H}_2(\mathfrak{y}_1\|\mathfrak{y}_2)$
4 :  $c \leftarrow (\mathfrak{y}_1, c_1)$
5 :  Return $(k, c)$
```

```
Encaps'_3(pk)
1 :  $r \hookleftarrow\$ \{0,1\}^{\lambda_1}$
2 :  $\alpha \leftarrow \mathsf{H}_0(r), k \hookleftarrow\$ \{0,1\}^{\lambda}$
3 :  $c_1 \leftarrow r \oplus \mathsf{H}_2(\tau(\alpha)\|\tau(\alpha x))$
4 :  $c \leftarrow (\tau(\alpha), c_1)$
5 :  Return $(k, c)$
```

Figure 2: Comparison between the oracles Encaps'_i for PSEC-KEM, which is used in Hybrid H_i for each $i \in \{0,1,2,3\}$, where the highlighted parts denote the difference from the previous hybrid.

  - Initialize the list $\mathcal{L} = \{(\tau(1), 0, 1), (\mathsf{pk} = \tau(x), 1, 0)\}$ and the vector of unknowns $\mathbf{x} \coloneqq (x_1)$ (for now, $\dim \mathbf{x} = 1$) where x_0 is the unknown (secret key) that maps to pk, i.e., $\tau(x_1) = \mathsf{pk}$. (Note: $x_1 = x$ but $\mathcal{D}$ does not know sk and simply name it as x_1 as an unknown.) $\mathcal{D}$ also initializes $\mathsf{H}^{\mathsf{resp}}_i = \{\}$ for $i = 0, 1, 2$, where $\mathsf{H}^{\mathsf{resp}}_i$ stores the random oracle queries of H_i.
  - Whenever $\mathcal{D}$ submits a query w to the RO H_i for $i = 0, 1, 2$:
    - If there is a pair $(w, R) \in \mathsf{H}^{\mathsf{resp}}_i$ for some string R, then return R.
    - Otherwise, select a uniformly random $R \hookleftarrow\$ \mathsf{codomain}(H_i)$ and add (w, R) to the set $\mathsf{H}^{\mathsf{resp}}_i$.
    - If w has the form $w = (\tau(\alpha)\|\cdot)$ for some α and if $\tau(\alpha)$ does not appear in $\mathcal{L}$, then update the vector $\mathbf{x} \leftarrow \mathbf{x} \circ \alpha$ and update all of the entries in $\mathcal{L}$ from $(\mathfrak{y}, \mathbf{a}, b)$ to $(\mathfrak{y}, \mathbf{a} \circ 0, b)$. Finally, add a new tuple $(\tau(\alpha), \mathbf{0} \circ 1, 0)$ to $\mathcal{L}$.
  - Whenever $\mathcal{D}$ submits a query $\mathfrak{y}_1, \mathfrak{y}_2$ to $\mathtt{Mult}(\cdot,\cdot)$:
    - If there exist tuples $(\mathfrak{y}_1, \mathbf{a}_1, b_1), (\mathfrak{y}_2, \mathbf{a}_2, b_2) \in \mathcal{L}$ for both $\mathfrak{y}_1$ and $\mathfrak{y}_2$, then add the tuple $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{a}_1 + \mathbf{a}_2, b_1 + b_2)$ to the list $\mathcal{L}$.
    - If $(\mathfrak{y}_1, \mathbf{a}_1, b_1) \in \mathcal{L}$ but $\mathfrak{y}_2$ does not appear in $\mathcal{L}$ and if $\mathfrak{y}_2 \in \tau(\mathbb{Z}_p)$, then first update $\mathbf{x} \leftarrow \mathbf{x} \circ y_2$ where y_2 is an unknown value that satisfies $\tau(y_2) = \mathfrak{y}_2$ and update all of the entries of the list $\mathcal{L}$ from $(\mathfrak{y}, \mathbf{a}, b)$ to $(\mathfrak{y}, \mathbf{a} \circ 0, b)$. Then add tuples $(\mathfrak{y}_2, \mathbf{0} \circ 1, 0)$ and $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{a}_1 \circ 0 + \mathbf{0} \circ 1, b_1)$ to $\mathcal{L}$. Do it similarly if $\mathfrak{y}_2$ appears in $\mathcal{L}$ and $\mathfrak{y}_1$ does not.
    - If both $\mathfrak{y}_1, \mathfrak{y}_2$ does not appear in $\mathcal{L}$, and if both $\mathfrak{y}_1, \mathfrak{y}_2 \in \tau(\mathbb{Z}_p)$, then first update $\mathbf{x} \leftarrow \mathbf{x} \circ y_1 \circ y_2$ where y_1 and y_2 are unknown values such that $\tau(y_i) = \mathfrak{y}_i$ for $i = 1, 2$, and update all the entries of the list $\mathcal{L}$ from $(\mathfrak{y}, \mathbf{a}, b)$ to $(\mathfrak{y}, \mathbf{a} \circ 0^2, b)$. Then add tuples $(\mathfrak{y}_1, \mathbf{0} \circ 1 \circ 0, 0)$, $(\mathfrak{y}_2, \mathbf{0} \circ 0 \circ 1, 0)$, and $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{0} \circ 1 \circ 1, 0)$ to $\mathcal{L}$.

- Whenever $\mathcal{D}$ submits a query $\mathfrak{y}$ to $\mathtt{Inv}(\cdot)$:
  - If there exists a tuple $(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L}$, then we adds the tuple $(\mathtt{Inv}(\mathfrak{y}), -\mathbf{a}, -b)$ to the list $\mathcal{L}$.
  - If $\mathfrak{y}$ does not appear in $\mathcal{L}$ and $\mathfrak{y} \in \tau(\mathbb{Z}_p)$, then first update $\mathbf{x} \leftarrow \mathbf{x} \circ y$ where y is an unknown value that satisfies $\tau(y) = \mathfrak{y}$ and update all of the entries of the list $\mathcal{L}$ from $(\mathfrak{y}, \mathbf{a}, b)$ to $(\mathfrak{y}, \mathbf{a} \circ 0^2, b)$. Then add tuples $(\mathfrak{y}, \mathbf{0} \circ 1, 0)$ and $(\mathtt{Inv}(\mathfrak{y}), \mathbf{0} \circ (-1), 0)$ to $\mathcal{L}$.
- Whenever $\mathcal{D}$ makes a query to O, $\mathcal{D}$ gets some bitstring $\mathfrak{y} \in \tau(\mathbb{Z}_p)$. If $\mathfrak{y}$ does not appear in $\mathcal{L}$ and $\dim(\mathbf{x}) = j$, then the challenger updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1}$ (i.e., increase the dimension of $\mathbf{x}$ by 1 and add the element x_{j+1} at the end), where x_{j+1} denotes an unknown variable satisfying the constraint $\tau(x_{j+1}) = \mathfrak{y}$, and updates all of the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$. Then the challenger adds $(\mathfrak{y}, \mathbf{0} \circ 1, 0)$ to $\mathcal{L}$.
- Whenever $\mathcal{D}$ makes a query to Encaps'_0 or Encaps'_1, simulating those oracles include queries to random oracles and the oracle O. Then $\mathcal{D}$ maintains the list $\mathcal{L}$ and the vector of unknowns $\mathbf{x}$ as described before per each oracle.

- Once $\mathcal{D}$ makes a query of the form $(\tau(\alpha_i) \| \tau(\alpha_i x))$ to H_2, parse $\tau(\alpha)$ and $\tau(\alpha x)$. Look up the list $\mathcal{L}$ and find j such that $x_j = \alpha_i$ (this is possible since $\mathcal{D}$ has been maintaining both the list $\mathcal{L}$ and the vector $\mathbf{x} = (x_1, x_2, \ldots, x_j, \ldots)$). We now have two cases:
  - If $(\tau(\alpha_i), \hat{u}_j, 0) \in \mathcal{L}$ where $\hat{u}_j$ denotes a unit vector with j^{th} element 1, and $(\tau(\alpha_i x), \mathbf{a}, b) \in \mathcal{L}$ for some $\mathbf{a}, b$, then $\mathcal{D}$ wins the quadratic bridge-finding game by outputting $(1, j, \mathbf{a}, b)$, and
  - If $(\tau(\alpha_i), \hat{u}_j, 0) \in \mathcal{L}$ but $\tau(\alpha_i x)$ does not appear in $\mathcal{L}$, then this implies that $\alpha_i x$ is a fresh unknown value. Increase $\dim \mathbf{x}$ by 1 and update all the entries in $\mathcal{L}$ from $(\mathfrak{y}, \mathbf{a}, b)$ to $(\mathfrak{y}, \mathbf{a} \circ 0, b)$. Add a new tuple $(\tau(\alpha_i x), \mathbf{0} \circ 1, 0)$ to $\mathcal{L}$ and $\mathcal{D}$ can win the quadratic bridge-finding game by outputting $(1, j, \mathbf{0} \circ 1, 0)$.

This proves that $\mathcal{D}$ can always win the quadratic bridge-finding game once $\mathcal{D}$ makes a query $\mathsf{H}_2(\tau(\alpha_i) \| \tau(\alpha_i x))$.

In Case (2), let this event Lucky. Since $\mathsf{H}_2(\tau(\alpha_i) \| \tau(\alpha_i x))$ was not queried, $c_{1,i}$ is basically random and unrelated to r_i. Hence, the adversary cannot have extra information to guess r_i from the information of (k_i, c_i) for $1 \le i \le q_{\mathsf{E}}$. Since there are q_{E} different index i and every random oracle query is matched with $\mathsf{H}_0(r_i)$ with probability at most $1/2^{\lambda_1}$, we observe that

$$\Pr[\mathsf{Lucky}] \le \frac{q_{\mathsf{H}} \cdot q_{\mathsf{E}}}{2^{\lambda_1}}.$$

Combining both cases with Lemma D.1, we have

$$\begin{aligned}
&\left|\Pr[\mathcal{D}^{H_0} = 1] - \Pr[\mathcal{D}^{H_1} = 1]\right| \\
&\quad \le \Pr[\mathsf{QDBridgeChal}^{\tau}_{\mathcal{D}}(\lambda) = 1] + \Pr[\mathsf{Lucky}] \\
&\quad \le \frac{3(q_{\mathsf{G}} + q_O)^2 + 5(q_{\mathsf{G}} + q_O) + 4}{2p - 2(3q_{\mathsf{G}} + q_O + 1)^2 - 2(2q_{\mathsf{G}} + q_O)} + \frac{q_{\mathsf{H}} \cdot q_{\mathsf{E}}}{2^{\lambda_1}} \\
&\quad \le \frac{3(q_{\mathsf{G}} + 2q_{\mathsf{E}})^2 + 5(q_{\mathsf{G}} + 2q_{\mathsf{E}}) + 4}{2p - 2(3q_{\mathsf{G}} + 2q_{\mathsf{E}} + 1)^2 - 4(q_{\mathsf{G}} + q_{\mathsf{E}})} + \frac{q_{\mathsf{H}} \cdot q_{\mathsf{E}}}{2^{\lambda_1}},
\end{aligned}$$

since $q_O \le 2q_{\mathsf{E}}$.

Indistinguishability of H_1 and H_2. We first observe that two hybrids are perfectly indistinguishable unless the random oracle query $\mathsf{H}_1(r_i)$ is queried for some round $i \le q_{\mathsf{E}}$. Let bad_i be the event that the random oracle query $\mathsf{H}_1(r_i)$ is queried in round i. Unless the random oracle query $\mathsf{H}_2(\mathfrak{y}_{1,i} \| \mathfrak{y}_{2,i})$ was made, we can view r_i as chosen uniformly at random from $\{0,1\}^{\lambda_1}$ (since otherwise, one can retrieve $r_i \leftarrow c_{1,i} \oplus \mathsf{H}_2(\mathfrak{y}_{1,i} \| \mathfrak{y}_{2,i})$). Let Z be the event that the random oracle query $\mathsf{H}_2(\mathfrak{y}_{1,i} \| \mathfrak{y}_{2,i})$ was made. Then we have

$$\begin{aligned}
\left|\Pr[\mathcal{D}^{H_1} = 1] - \Pr[\mathcal{D}^{H_2} = 1]\right| &\le \sum_{i=1}^{q_{\mathsf{E}}} \Pr[\mathsf{bad}_i] \\
&\le \sum_{i=1}^{q_{\mathsf{E}}} \left\{ \Pr[Z] + \Pr[\mathsf{bad}_i | \overline{Z}] \right\} \\
&\le q_{\mathsf{E}} \left(\frac{q_{\mathsf{H}}}{p} + \frac{q_{\mathsf{H}}}{2^{\lambda_1}} \right).
\end{aligned}$$

Indistinguishability of H_2 and H_3. Analyzing the distinguishing probability between the hybrid H_2 and H_3 works similar as the indistinguishability of H_0 and H_1 since the difference between Encaps'_2 and Encaps'_3 is the same as the difference between Encaps'_1 and Encaps'_0. Hence,

$$\begin{aligned}
&\left|\Pr[\mathcal{D}^{H_2} = 1] - \Pr[\mathcal{D}^{H_3} = 1]\right| \\
&\quad \le \frac{3(q_{\mathsf{G}} + 2q_{\mathsf{E}})^2 + 5(q_{\mathsf{G}} + 2q_{\mathsf{E}}) + 4}{2p - 2(3q_{\mathsf{G}} + 2q_{\mathsf{E}} + 1)^2 - 4(q_{\mathsf{G}} + q_{\mathsf{E}})} + \frac{q_{\mathsf{H}} \cdot q_{\mathsf{E}}}{2^{\lambda_1}}.
\end{aligned}$$

Putting everything together, we have

$$\begin{aligned}
\Pr\left[\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda) = 1\right] &\le \frac{1}{2} + \left|\Pr[\mathcal{D}^{H_0} = 1] - \Pr[\mathcal{D}^{H_3} = 1]\right| \\
&\le \frac{1}{2} + \sum_{i=0}^{2} \left|\Pr[\mathcal{D}^{H_i} = 1] - \Pr[\mathcal{D}^{H_{i+1}} = 1]\right| \\
&\le \frac{1}{2} + \frac{3(q_{\mathsf{G}} + 2q_{\mathsf{E}})^2 + 5(q_{\mathsf{G}} + 2q_{\mathsf{E}}) + 4}{p - (3q_{\mathsf{G}} + 2q_{\mathsf{E}} + 1)^2 - 2(q_{\mathsf{G}} + q_{\mathsf{E}})} + \frac{3q_{\mathsf{H}} q_{\mathsf{E}}}{2^{\lambda_1}} + \frac{q_{\mathsf{H}} q_{\mathsf{E}}}{p}. \quad \square
\end{aligned}$$

E Formal Description of Several Security Games

E.1 Formal Description of the CPA Indistinguishability Game for a KEM

The Generic CPA Indistinguishability Game for a KEM. Let $\Pi = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ be a KEM and $\mathcal{A}$ be an adversary attacking Π. We define the following CPA indistinguishability game in the ROM+GGM, in which the attacker tries to distinguish between the real key k and a random key.

The Generic CPA Indistinguishability Game $\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa/cca}}_{\mathcal{A},\Pi}(\lambda)$:

1. The challenger runs $\mathsf{Gen}(1^\lambda)$ to obtain a public key pk and a secret key sk. Here, sk is chosen randomly from the group $\mathbb{Z}_p$ where p is a 2λ-bit prime, and $\mathsf{pk} = \tau(\mathsf{sk})$.
2. The challenger invokes $\mathsf{Encaps}(\mathsf{pk}, 1^\lambda)$ to obtain a key-ciphertext pair (k, c).

3. The challenger chooses a uniform bit $b \in \{0,1\}$. If $b = 0$, set $\hat{k} \coloneqq k$; if $b = 1$, choose a random $\hat{k} \leftarrow\!\$ \{0,1\}^\lambda$.
4. $\mathcal{A}$ gets a tuple $(\mathsf{pk}, \hat{k}, c)$. The adversary is given access to the oracle $\mathsf{Encaps}_b(\cdot)$, the random oracles H, and the generic group oracle $(\mathtt{Mult}(\cdot,\cdot), \mathtt{Inv}(\cdot))$. Here, $\mathsf{Encaps}_b(\mathsf{pk})$ works as follows:

$$\mathsf{Encaps}_b(\mathsf{pk}) \coloneqq \begin{cases} \mathsf{Encaps}(\mathsf{pk}, 1^\lambda) & \text{if } b = 0, \\ (R \leftarrow\!\$ \{0,1\}^\lambda, \mathsf{Encaps}(\mathsf{pk}, 1^\lambda).\mathsf{ctxt}) & \text{if } b = 1. \end{cases}$$

After multiple queries, $\mathcal{A}$ outputs a bit b'.
5. $\mathcal{A}$ wins if $b' = b$, which we write $\mathsf{KEM}^{\tau,\mathsf{H},\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda) = 1$, and the output of the game is 0 if $\mathcal{A}$ fails to guess b correctly.

E.2 Formal Description of the Quadratic Bridge-Finding Game

The Generic Quadratic Bridge-Finding Game $\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda)$ with a Bit-Fixing Attacker $\mathcal{A} = \left(\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{pre}}, \mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{on}}\right)$:

Preprocessing Phase:

1. $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{pre}}$ takes as input $\tau(1)$. Here, $\tau : \mathbb{Z}_p \to \mathbb{G}$ is a labeling map where $\mathbb{G}$ is a set of bitstrings of length $\ell \geq \log p$, and $p > 2^{2\lambda}$ is a prime.
2. $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{pre}}$ fixes P input/output pairs of the map $\tau : \mathbb{Z}_p \to \mathbb{G}$, i.e., $(t_1, \tau(t_1)), \ldots, (t_P, \tau(t_P))$. (Note: the rest of the map τ is chosen uniformly at random as long as it remains to be injective.)

Online Phase:

1. The challenger receives P pre-fixed points, pick $x \leftarrow\!\$ \mathbb{Z}_p$ and initializes the list $\mathcal{L} = \{(\tau(1), 0, 1), (\tau(x), 1, 0), (\tau(t_1), 0, t_1), \ldots, (\tau(t_P), 0, t_P)\}$.
2. The adversary $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{on}}$ is given $\tau(1)$ and $\tau(x)$ as input. And $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{on}}$ is additionally given P pre-fixed points of the map τ from $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{pre}}$.
3. The challenger maintains a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ which is a vector of unknowns such that $x_1 = x$, where the dimension n satisfies the following:
   - initially, $n = 1$ since the only unknown is x, and
   - the number of unknowns n increases by at most 2 as $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{on}}$ makes query to the generic group oracle $(\mathtt{Mult}, \mathtt{Inv})$ or a generic oracle O that selects a new random unknown variable $x_j \leftarrow\!\$ \mathbb{Z}_p$ and outputs $\tau(x_j)$.
4. $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{on}}$ is allowed to query the generic group oracle $(\mathtt{Mult}(\cdot,\cdot), \mathtt{Inv}(\cdot))$ and the oracle O.
   (a) If $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{on}}$ submits a query to O, $\mathcal{A}$ gets some bitstring $\mathfrak{y} \in \tau(\mathbb{Z}_p)$. If $\mathfrak{y}$ does not appear in $\mathcal{L}$ and $\dim(\mathbf{x}) = j$, then the challenger updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1}$ (i.e., increase the dimension of $\mathbf{x}$ by 1 and add the element x_{j+1} at the end), where x_{j+1} denotes an unknown variable satisfying the constraint $\tau(x_{j+1}) = \mathfrak{y}$, and updates all of the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$. Then the challenger adds $(\mathfrak{y}, \mathbf{0} \circ 1, 0)$ to $\mathcal{L}$.
   (b) If $\mathcal{A}^{\mathsf{BF\text{-}GG}(P)}_{\mathsf{on}}$ submits a query $\mathfrak{y}_1, \mathfrak{y}_2$ to $\mathtt{Mult}(\cdot,\cdot)$:
   - If there exist tuples $(\mathfrak{y}_1, \mathbf{a}_1, b_1), (\mathfrak{y}_2, \mathbf{a}_2, b_2) \in \mathcal{L}$ for both $\mathfrak{y}_1$ and $\mathfrak{y}_2$, then the challenger adds the tuple $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{a}_1 + \mathbf{a}_2, b_1 + b_2)$ to the list $\mathcal{L}$.
   - If $(\mathfrak{y}_1, \mathbf{a}_1, b_1) \in \mathcal{L}$ but $\mathfrak{y}_2$ does not appear in $\mathcal{L}$ and if $\mathfrak{y}_2 \in \tau(\mathbb{Z}_p)$, and if $\dim(\mathbf{x}) = j$, then the challenger first updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1}$ where x_{j+1} denotes an unknown variable satisfying the constraint $\tau(x_{j+1}) = \mathfrak{y}_2$ and updates all of the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$. Then the challenger adds tuples $(\mathfrak{y}_2, \mathbf{0} \circ 1, 0)$ and $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{a}_1 \circ 0 + \mathbf{0} \circ 1, b_1)$ to $\mathcal{L}$. Do it similarly if $\mathfrak{y}_2$ appears in $\mathcal{L}$ and $\mathfrak{y}_1$ does not.
   - If both $\mathfrak{y}_1$ and $\mathfrak{y}_2$ does not appear in $\mathcal{L}$, and if both $\mathfrak{y}_1, \mathfrak{y}_2 \in \tau(\mathbb{Z}_p)$, and if $\dim(\mathbf{x}) = j$, then the challenger first updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1} \circ x_{j+2}$ where x_{j+1} and x_{j+2} are unknown values such that $\tau(x_{j+i}) = \mathfrak{y}_i$ for $i = 1, 2$, and updates all the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0^2, b)$. Then the challenger adds tuples $(\mathfrak{y}_1, \mathbf{0} \circ 1 \circ 0, 0)$, $(\mathfrak{y}_2, \mathbf{0} \circ 0 \circ 1, 0)$, and $(\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2), \mathbf{0} \circ 1 \circ 1, 0)$ to $\mathcal{L}$.
   (c) If $\mathcal{A}$ submits a query $\mathfrak{y}$ to $\mathtt{Inv}(\cdot)$:
   - If there exists a tuple $(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L}$, then the challenger adds the tuple $(\mathtt{Inv}(\mathfrak{y}), -\mathbf{a}, -b)$ to the list $\mathcal{L}$.
   - If $\mathfrak{y}$ does not appear in $\mathcal{L}$ and $\mathfrak{y} \in \tau(\mathbb{Z}_p)$, and if $\dim(\mathbf{x}) = j$, then the challenger first updates $\mathbf{x} \leftarrow \mathbf{x} \circ x_{j+1}$ where x_{j+1} is an unknown value that satisfies $\tau(x_{j+1}) = \mathfrak{y}$ and updates all of the entries of the list $\mathcal{L}$ from $(\mathfrak{s}, \mathbf{a}, b)$ to $(\mathfrak{s}, \mathbf{a} \circ 0, b)$. Then the challenger adds tuples $(\mathfrak{y}, \mathbf{0} \circ 1, 0)$ and $(\mathtt{Inv}(\mathfrak{y}), \mathbf{0} \circ (-1), 0)$ to $\mathcal{L}$.
5. If at any point in time $\mathcal{A}$ outputs a tuple $(i_1, i_2, \mathbf{a}, b)$ such that (1) $1 \leq i_1, i_2 \leq \dim(\mathbf{x})$, (2) $(\mathbf{a}, b) \in \mathbb{Z}_p^{\dim(\mathbf{x})} \times \mathbb{Z}_p$, and (3) $x_{i_1} x_{i_2} = \mathbf{a} \cdot \mathbf{x} + b \mod p$, then we say that the event QDBridge occurs. In this case, the output of the game is 1 and we write $\mathsf{QDBridgeChal}^{\tau}_{\mathcal{A}}(\lambda) = 1$. If QDBridge never occurs, then the output of the game is 0.

F The Quadratic Discrete-Log Game

The Quadratic Discrete-Log Game. We define a variant of the discrete-log game called *the quadratic discrete-log game* which might be crucial in analyzing the security of various cryptographic primitives or protocols.

The Generic Quadratic Discrete-Log Game $\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda)$:

1. The adversary $\mathcal{A}$ is given $(\mathfrak{g} = \tau(1), \tau(x))$ for a random value of $x \in \mathbb{Z}_p$. Here, $\tau : \mathbb{Z}_p \to \mathbb{G}$ is a labeling map where $\mathbb{G}$ is a set of bitstrings of length $\ell \geq \log p$. Here, p is a 2λ-bit prime.
2. $\mathcal{A}$ is allowed to query the usual generic group oracles $(\mathtt{Mult}, \mathtt{Inv})$ and is additionally allowed to query $\mathtt{DLog}(\tau(y))$, but *only* if $\tau(y)$ is "fresh", i.e., $\tau(y)$ is not $\tau(x)$, and $\tau(y)$ has not been the output of a previous random generic group query.
3. After multiple queries, $\mathcal{A}$ outputs a tuple $(a, b, \mathfrak{y})$ where $a, b \in \mathbb{Z}_p$ with $a \neq 0$ and $\mathfrak{y} \in \mathbb{G}$.
4. The output of the game is defined to be $\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda) = 1$ if $\mathfrak{y} = \tau(ax^2 + bx)$, and 0 otherwise.

Lemma F.1. *The probability the attacker making at most* q_G *generic group oracle queries wins the generic quadratic discrete-log game* $\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda)$ *(even with access to the restricted* $\mathtt{DLog}$ *oracle) is at most*

$$\Pr[\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda) = 1] \leq \frac{3q_\mathsf{G}^2 + 7q_\mathsf{G} + 4}{2p - 2(3q_\mathsf{G} + 2)^2},$$

in the generic group model of prime order p, where the randomness is taken over the selection of the labeling map τ, the challenge x, as well as any random coins of $\mathcal{A}$.

Proof. The proof largely borrows the same idea from the prior work [6, Lemma 1]. Intuitively, the proof works by maintaining a list $\mathcal{L}$ of tuples $(\tau(y), a, b)$ where $\tau(y) \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$ such that $y = ax + b$ for every oracle output $\tau(y)$. Initially, the list $\mathcal{L}$ contains two items $(\tau(1), 0, 1)$ and $(\tau(x), 1, 0)$. $\mathcal{L}$ is updated after every query to the generic group oracles as follows:

- if $(\tau(y_1), a_1, b_1) \in \mathcal{L}$ and $(\tau(y_2), a_2, b_2) \in \mathcal{L}$, then querying $\mathtt{Mult}(\tau(y_1), \tau(y_2))$ will result in adding a tuple $(\tau(y_1 + y_2), a_1 + a_2, b_1 + b_2)$ to $\mathcal{L}$, and
- if $(\tau(y), a, b) \in \mathcal{L}$ then querying $\mathtt{Inv}(\tau(y))$ will result in adding a tuple $(-\tau(y), -a, -b)$ to $\mathcal{L}$.

We say that the event BRIDGE occurs if $\mathcal{L}$ contains two tuples $(\tau(y_1), a_1, b_1)$ and $(\tau(y_2), a_2, b_2)$ such that $\tau(y_1) = \tau(y_2)$ but $(a_1, b_1) \neq (a_2, b_2)$. We can further use the restricted discrete-log oracle $\mathtt{DLog}$ to maintain the invariant that every output of the generic group oracles $\mathtt{Mult}$ and $\mathtt{Inv}$ can be added to $\mathcal{L}$. In particular, if we ever encounter an input $\mathfrak{y} = \tau(b)$ that *does not* already appear in $\mathcal{L}$, then $\mathfrak{y}$ is considered *fresh* and we can simply query $\mathtt{DLog}$ to extract $b = \mathtt{DLog}(\mathfrak{y})$, ensuring that the tuple $(\mathfrak{y}, 0, b)$ is added to $\mathcal{L}$ before the generic group query is processed.

If the event BRIDGE occurs, it becomes trivial for the attacker to find $\mathfrak{y}$ that satisfies $\mathfrak{y} = ax^2 + bx$ for some $a, b \in \mathbb{Z}_p$ with $a \neq 0$ because the attacker can indeed extract the secret x, i.e., if $\mathcal{L}$ contains two tuples $(\tau(y_1), a_1, b_1)$ and $(\tau(y_2), a_2, b_2)$ such that $\tau(y_1) = \tau(y_2)$ but $(a_1, b_1) \neq (a_2, b_2)$, then we know that $a_1x + b_1 = a_2x + b_2$ with $a_1 \neq a_2$ (since otherwise we have $b_1 = b_2$ which contradicts the fact that $(a_1, b_1) \neq (a_2, b_2)$) which allows us to solve for $x = (b_2 - b_1)(a_1 - a_2)^{-1}$.

If the event BRIDGE does not occur then after all queries have finished the attacker can still view x as an element yet to be sampled from a uniform distribution over a set of size at least $p - |\mathcal{L}|^2$. To see this, note that each tuple of distinct elements $(\tau(y_1), a_1, b_1)$ and $(\tau(y_2), a_2, b_2)$ eliminates at most one possible value of x, i.e., since $\tau(y_1)$ and $\tau(y_2)$ are distinct we have $x \neq (b_2 - b_1)(a_1 - a_2)^{-1}$. We further observe that $|\mathcal{L}| \leq 3q_\mathsf{G} + 2$ since each query to the generic group oracles adds at most 3 elements to $\mathcal{L}$ — equality holds when both inputs to $\mathtt{Mult}(\cdot, \cdot)$ are fresh. Now, the attacker wins if and only if s/he outputs a tuple $(a, b, \mathfrak{y})$ with nonzero a and $\mathfrak{y} = \tau(ax^2 + bx)$. Since we ensured that $\mathfrak{y}$ appears in $\mathcal{L}$, we have that $\mathfrak{y} = \tau(a'x + b')$ for some known $a', b' \in \mathbb{Z}_p$, which implies that $ax^2 + bx = a'x + b' \mod p$ where $a \neq 0$. Since this quadratic equation has at most 2 roots in $\mathbb{Z}_p$ there are at most 2 out of at least $p - (3q_\mathsf{G} + 2)^2$ remaining choices of x which satisfy the equation $ax^2 + bx = a'x + b' \mod p$. Thus, we have that

$$\Pr\left[\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda) = 1 \mid \overline{\mathsf{BRIDGE}}\right] \leq \frac{2}{p - |\mathcal{L}|^2} \leq \frac{2}{p - (3q_\mathsf{G} + 2)^2}.$$

Next, we have that $\Pr[\mathsf{BRIDGE}] \leq \frac{3q_\mathsf{G}^2 + 7q_\mathsf{G}}{2p - 2(3q_\mathsf{G} + 2)^2}$ from [6, Lemma 1]. Hence, the probability the attacker succeeds is at most

$$\begin{aligned}
&\Pr[\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda) = 1] \\
&\leq \Pr[\mathsf{BRIDGE}] + \Pr[\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda) = 1 \mid \overline{\mathsf{BRIDGE}}] \\
&\leq \frac{3q_\mathsf{G}^2 + 7q_\mathsf{G}}{2p - 2(3q_\mathsf{G} + 2)^2} + \frac{2}{p - (3q_\mathsf{G} + 2)^2} \\
&= \frac{3q_\mathsf{G}^2 + 7q_\mathsf{G} + 4}{2p - 2(3q_\mathsf{G} + 2)^2}.
\end{aligned}$$

□

The Quadratic Discrete-Log Game with Preprocessing. Now we consider a preprocessing setting in a similar manner. We first define the quadratic discrete-log game in the *bit-fixing* generic group model which is described formally below.

The Generic Quadratic Discrete-Log Game $\mathsf{QDLogChal}^{\tau}_{\mathcal{A}}(\lambda)$ with a Bit-Fixing Preprocessing Attacker $\mathcal{A} \coloneqq (\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)})$:

Preprocessing Phase:

1. $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ takes as input $\mathfrak{g} = \tau(1)$.
2. $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ fixes P input/output pairs of the map $\tau : \mathbb{Z}_p \to \mathbb{G}$, i.e., $(t_1, \tau(t_1)), \ldots, (t_P, \tau(t_P))$. (Note: the rest of the map τ is chosen uniformly at random as long as it remains to be injective.)

Online Phase:

1. The adversary $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ is given $\mathfrak{g} = \tau(1)$ and $\tau(x)$ for a random value of $x \in \mathbb{Z}_p$.
2. $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ is allowed to query the usual generic group oracles $(\mathtt{Mult}, \mathtt{Inv})$ and is additionally allowed to query $\mathtt{DLog}(\tau(y))$, but *only* if $\tau(y)$ is "fresh", i.e., $\tau(y)$ is not

$\tau(x)$, and $\tau(y)$ has not been the output of a previous random generic group query.

3. After multiple queries, $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ outputs a tuple $(a, b, \mathfrak{y})$ where $a, b \in \mathbb{Z}_p$ with $a \neq 0$ and $\mathfrak{y} \in \mathbb{G}$.
4. The output of the game is defined to be $\mathsf{QDLogChal}_{\mathcal{A}}^{\tau}(\lambda) = 1$ if $\mathfrak{y} = \tau(ax^2 + bx)$, and 0 otherwise.

Lemma F.2. *Let $p > 2^{2\lambda}$ be a prime number and let $\mathcal{A} := \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ fixes P input/output pairs of the labeling map and $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} := q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles. Then*

$$\Pr[\mathsf{QDLogChal}_{\mathcal{A}}^{\tau}(\lambda) = 1] \leq \frac{3(q_{\mathsf{G}}^{\mathsf{on}})^2 + 5q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{G}}^{\mathsf{on}}P + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + 2) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + 2)^2},$$

in the generic group model of prime order p, where the randomness is taken over the selection of the labeling map τ, the challenge x, as well as any random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$.

Proof. The proof works largely the same as Lemma F.1, except for the fact that

- the upper bound of the size of the list is increased by P: $|\mathcal{L}| \leq P + 3q_{\mathsf{G}}^{\mathsf{on}} + 2$,
- if the event BRIDGE does not occur then after all queries have finished the attacker can still view x as an element yet to be sampled from a uniform distribution over a set of size at least $p - (|\mathcal{L}|^2 - P^2)$, and
- the probability of the event BRIDGE becomes higher, i.e., $\Pr[\mathsf{BRIDGE}] \leq \frac{1.5(q_{\mathsf{G}}^{\mathsf{on}})^2 + 2.5q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{G}}^{\mathsf{on}}P}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}}+2) - (3q_{\mathsf{G}}^{\mathsf{on}}+2)^2}$ (see Lemma 4.3).

Hence, we have

$$\begin{aligned}\Pr&\left[\mathsf{QDLogChal}_{\mathcal{A}}^{\tau}(\lambda) = 1 \mid \overline{\mathsf{BRIDGE}}\right] \\ &\leq \frac{2}{p - (|\mathcal{L}|^2 - P^2)} \\ &\leq \frac{2}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + 2) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2)^2},\end{aligned}$$

and

$$\begin{aligned}&\Pr[\mathsf{QDLogChal}_{\mathcal{A}}^{\tau}(\lambda) = 1] \\ &\leq \Pr[\mathsf{BRIDGE}] + \Pr[\mathsf{QDLogChal}_{\mathcal{A}}^{\tau}(\lambda) = 1 \mid \overline{\mathsf{BRIDGE}}] \\ &\leq \frac{1.5(q_{\mathsf{G}}^{\mathsf{on}})^2 + 2.5q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{G}}^{\mathsf{on}}P}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + 2) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2)^2} + \frac{2}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + 2) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2)^2} \\ &= \frac{3(q_{\mathsf{G}}^{\mathsf{on}})^2 + 5q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{G}}^{\mathsf{on}}P + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + 2) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + 2)^2}.\end{aligned}$$

□

An immediate corollary of Lemma F.2 can be obtained by applying [11, Theorem 1] and by setting an optimal value of $P \approx Sq_{\mathsf{G}}^{\mathsf{on}}$ and $\gamma = 2^{-2\lambda}$.

Corollary F.3. *Let $p > 2^{2\lambda}$ be a prime number and $N \in \mathbb{N}$ be a parameter. Let $\mathcal{A} := \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{AI\text{-}GG}}, \mathcal{A}_{\mathsf{on}}^{\mathsf{AI\text{-}GG}}\right)$ be a pair of auxiliary-input generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{AI\text{-}GG}}$ outputs an S-bit hint after looking at the entire map τ and $\mathcal{A}_{\mathsf{on}}^{\mathsf{AI\text{-}GG}}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} := q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles. Then*

$$\begin{aligned}\Pr&\left[\mathsf{QDLogChal}_{\mathcal{A}}^{\tau}(\lambda) = 1\right] \\ &\leq \frac{3(q_{\mathsf{G}}^{\mathsf{on}})^2 + 5q_{\mathsf{G}}^{\mathsf{on}} + 2S(q_{\mathsf{G}}^{\mathsf{on}})^2 + 4}{p - 2Sq_{\mathsf{G}}^{\mathsf{on}}(3q_{\mathsf{G}}^{\mathsf{on}} + 2) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2)^2} + 2^{-2\lambda+1}.\end{aligned}$$

G Missing Proofs

Reminder of Lemma 4.3. *Let $p > 2^{2\lambda}$ be a prime number and $N \in \mathbb{N}$ be a parameter. Let $\mathcal{A} := \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ fixes P input/output pairs of the labeling map τ and $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} := q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles. Then*

$$\Pr\left[\mathsf{BridgeChal}_{\mathcal{A}}^{\tau,N}(\lambda) = 1\right] \leq \frac{q_{\mathsf{G}}^{\mathsf{on}}(N + P) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}} + 1)/2}{p - 2P(N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1) - (N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 - N},$$

in the GGM of prime order p, where the randomness is taken over the selection of $x_1, \ldots, x_N, \tau$ as well as any random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$.

Proof of Lemma 4.3. The proof is similar to the proof of [6, Theorem 5]. Intuitively, we first consider the probability that the bridge event occurs conditioning on the event that the bridge event has not yet occurred until the previous query, i.e., let $\overline{\mathsf{BRIDGE}}_{<i}^{N}$ be the event that the event BRIDGE^N has not occurred until the $(i-1)^{th}$ query.

Before we even receive the output $\mathfrak{y}_i$, we already know the values $\mathbf{a}_i, b_i$ such that the tuple $(\mathfrak{y}_i, \mathbf{a}_i, b_i)$ will be added to $\mathcal{L}$. If $\mathcal{L}$ does already contain this *exact* tuple, then outputting $\mathfrak{y}_i$ will not produce the event BRIDGE^N. If $\mathcal{L}$ does not already contain this tuple $(\mathfrak{y}_i, \mathbf{a}_i, b_i)$, then we are interested in the event B_i that some other tuple $(\mathfrak{y}_i, \mathbf{a}_i', b_i')$ has been recorded with $(\mathbf{a}_i', b_i') \neq (\mathbf{a}_i, b_i)$. Observe that B_i occurs if and only if there exists a tuple of the form $(\cdot, \mathbf{a}, b)$ with $(\mathbf{a} - \mathbf{a}_i) \cdot \mathbf{x} = b_i - b$ and $(\mathbf{a}, b) \neq (\mathbf{a}_i, b_i)$. If we pick $\mathbf{x}$ randomly, the probability that $(\mathbf{a} - \mathbf{a}_i) \cdot \mathbf{x} = b_i - b$ would be $1/p$. However, we cannot quite view $\mathbf{x}$ as random due to the restrictions, i.e., because we condition on the event $\overline{\mathsf{BRIDGE}}_{<i}^{N}$ we know that for any distinct tuples $(\mathfrak{y}_j, \mathbf{a}_j, b_j)$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k)$ we know that $\mathbf{a}_j \cdot \mathbf{x} + b_j \neq \mathbf{a}_k \cdot \mathbf{x} + b_k$, i.e., $\mathfrak{y}_j \neq \mathfrak{y}_k$.

Consider sampling $\mathbf{x}$ uniformly at random subject to this restriction. Let $r \leq N$ be an index such that $\mathbf{a}[r] - \mathbf{a}_i[r] \neq 0$ and suppose that $x_r = \mathbf{x}[r]$ is the last value sampled. We also define two subsets $\mathcal{L}_0, \mathcal{L}_1 \subseteq \mathcal{L}$ where $\mathcal{L}_0 := \{(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L} : \mathbf{a} = \mathbf{0}\}$ and $\mathcal{L}_1 := \{(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L} : \mathbf{a} \neq \mathbf{0}\}$. Now we observe that $\mathcal{L}_0 \cup \mathcal{L}_1 = \mathcal{L}$ and if we pick two distinct pairs $(\mathfrak{y}_1, \mathbf{a}_1, b_1)$ and $(\mathfrak{y}_2, \mathbf{a}_2, b_2)$ both from $\mathcal{L}_0$ we are guaranteed to have that $\mathbf{a}_1 \cdot \mathbf{x} + b_1 \neq \mathbf{a}_2 \cdot \mathbf{x} + b_2$ since $\mathbf{a}_1 = \mathbf{a}_2 = \mathbf{0}$, which implies that we are guaranteed to not cause any bridge event.

At this point, we can view x_r as being drawn uniformly at random from a set of at least $p - (|\mathcal{L}|^2 - |\mathcal{L}_0|^2) - (N - 1)$ remaining values, subject to all of the restrictions. To see this, we observe the following:

- x_r must be distinct from $x_1, \dots, x_{r-1}$ since otherwise we would have had a bridge event, and
- each pair of distinct elements $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}$, we have the following cases:
  - if $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_1$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_1$, then this pair eliminates at most one possible value of x_r, i.e., since $\mathfrak{y}_j$ and $\mathfrak{y}_k$ are distinct, we have $\mathbf{a}_j \cdot \mathbf{x} + b_j \neq \mathbf{a}_k \cdot \mathbf{x} + b_k$, which implies that $x_r \neq \left((b_k - b_j) - \sum_{i=1, i\neq j}^{N} (\mathbf{a}_k[i] - \mathbf{a}_j[i]) x_i\right) (\mathbf{a}_j[r] - \mathbf{a}_k[r])^{-1}$,
  - if $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_1$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_0$, then this pair eliminates at most one possible value of x_r, i.e., since $\mathfrak{y}_j$ and $\mathfrak{y}_k$ are distinct, we have $\mathbf{a}_j \cdot \mathbf{x} + b_j \neq \mathbf{0} \cdot \mathbf{x} + b_k$, which implies that $x_r \neq \left((b_k - b_j) + \sum_{i=1, i\neq j}^{N} \mathbf{a}_j[i] x_i\right) \mathbf{a}_j[r]^{-1}$,
  - $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_0$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_1$, then this pair eliminates at most one possible value of x_r, i.e., since $\mathfrak{y}_j$ and $\mathfrak{y}_k$ are distinct, we have $\mathbf{0} \cdot \mathbf{x} + b_j \neq \mathbf{a}_k \cdot \mathbf{x} + b_k$, which implies that $x_r \neq \left((b_k - b_j) - \sum_{i=1, i\neq j}^{N} \mathbf{a}_k[i] x_i\right) (-\mathbf{a}_k[r])^{-1}$, and
  - if both $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_0$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_0$, then since both $\mathbf{a}_j = \mathbf{a}_k = \mathbf{0}$, it is a contradiction that $\mathbf{a}_j[r] - \mathbf{a}_k[r] \neq 0$ and therefore we do not eliminate any value of x_r.
- Hence, the total number of values that we remove when we draw x_r is at most $(r-1) + |\mathcal{L}_1|^2 + 2|\mathcal{L}_0||\mathcal{L}_1| \leq (N-1) + (|\mathcal{L}|^2 - |\mathcal{L}_0|^2)$.

We also observe that $|\mathcal{L}| \leq N + P + 3q_{\mathsf{G}}^{\mathsf{on}} + 1$ and $|\mathcal{L}_0| \geq P$ since the list already contains P additional pre-fixed pairs and each generic group oracle query adds *at most* three new tuples to $\mathcal{L}$ — exactly three in the case that we query `Mult`$(\mathfrak{y}_1, \mathfrak{y}_2)$ on two fresh elements. Thus, the probability that $(\mathbf{a} - \mathbf{a}_i) \cdot \mathbf{x} = b_i - b$ is at most $\frac{1}{p - (N+P+3q_{\mathsf{G}}^{\mathsf{on}}+1)^2 + P^2 - (N-1)}$. Union bounding over all tuples $(\cdot, \mathbf{a}, b) \in \mathcal{L}$, we have

$$\Pr\left[B_i : \overline{\mathsf{BRIDGE}_{<i}^N}\right] \leq \frac{N + P + 3i}{p - (N + P + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 + P^2 - N} = \frac{N + P + 3i}{p - 2P(N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1) - (N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 - N}.$$

To complete the proof, we observe that

$$\Pr\left[\mathsf{BridgeChal}_{\mathcal{A}}^{\tau,N}(\lambda) = 1\right] = \sum_{i \leq q_{\mathsf{G}}^{\mathsf{on}}} \Pr\left[B_i : \overline{\mathsf{BRIDGE}_{<i}^N}\right]$$
$$\leq \sum_{i \leq q_{\mathsf{G}}^{\mathsf{on}}} \frac{N + P + 3i}{p - 2P(N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1) - (N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 - N}$$
$$= \frac{q_{\mathsf{G}}^{\mathsf{on}}(N + P) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}} + 1)/2}{p - 2P(N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1) - (N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 - N}. \qquad \square$$

Reminder of Theorem 4.4. *Let $\Pi_{\setminus\{0\}} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ be a nonzero Schnorr signature scheme and $p > 2^{2\lambda}$ be a prime number. Let $N \in \mathbb{N}$ be a parameter and $\left(\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ fixes P_1 input/output pairs of a random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda_1}$ and P_2 input/output pairs of the map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $2P_1 + P_2 = P$, and the hint $\mathsf{str}_{\tau,\mathsf{H}}$ is only dependent on those P points. If $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} \coloneqq q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles, $q_{\mathsf{H}}^{\mathsf{on}}$ queries to the random oracle, and $q_{\mathsf{S}}^{\mathsf{on}}$ queries to the signing oracle, then*

$$\Pr\left[\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A}_{\mathsf{Sig.on},\mathsf{str}_{\tau,\mathsf{H}}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \Pi_{\setminus\{0\}}^{\mathsf{short}}}(\lambda) = 1\right] \leq \varepsilon,$$

with

$$\varepsilon = \frac{q_{\mathsf{G}}^{\mathsf{on}}(N + P) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}} + 1)/2}{p - 2P(N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1) - (N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 - N} + \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P)}{p} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P}{p - (N + P + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)} + \frac{2q_{\mathsf{H}}^{\mathsf{on}} + 1}{2^{\lambda_1}},$$

where the randomness is taken over the selection of τ and the random coins of $\mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$.

Proof of Theorem 4.4. Given a bit-fixing generic adversary $\left(\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{Sig.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ that attacks the short nonzero Schnorr signature scheme, we construct the following efficient generic algorithm $\mathcal{A}_{\mathsf{bridge}} = \left(\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{bridge.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ in the bit-fixing model which tries to succeed in the 1-out-of-N generic BRIDGE^N-finding game $\mathsf{BridgeChal}_{\mathcal{A}_{\mathsf{bridge}}}^{\tau,N}(\lambda)$:

Algorithm $\left(\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{bridge.on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$:

<u>Preprocessing Phase:</u>
The algorithm is given $p, \mathfrak{g} = \tau(1)$ as input.

1. $\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ simply runs $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ and outputs the same P_1 input/output pairs of H $(\{(h_1, \mathsf{H}(h_1)), \dots, (h_{P_1}, \mathsf{H}(h_{P_1}))\})$ and P_2 input/ output pairs of τ $(\{(t_1, \tau(t_1)), \dots, (t_{P_2}, \tau(t_{P_2}))\})$ as being fixed by $\mathcal{A}_{\mathsf{Sig.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$.
2. For each fixed RO input/output pair $(h_i, \mathsf{H}(h_i))$ where $i \in [P_1]$, $\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ first parses $h_i = I_i' \| m_i'$ where I_i' is a bitstring of length ℓ while m_i' represents the remaining part of the bitstring. If I_i' is a fresh ℓ-bit string, i.e., $I_i' \neq \tau(t_j)$ for all $j \in [P_2]$, then $\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ picks a value for $\tau^{-1}(I_i') \eqqcolon r_i'$ where the labeling map τ will be sampled later. (Note: this is because the attacker can query a restricted discrete-log oracle for a fresh bitstring but the labeling map τ has not been sampled.) Finally, $\mathcal{A}_{\mathsf{bridge.pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ also outputs the fixed input/output pairs of τ $\{(r_1', I_1'), \dots, (r_{P_1}', I_{P_1}')\}$.
3. Now the challenger initializes the list $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$, where
   - $\mathcal{L}_1 = \{(\tau(1), \mathbf{0}, 1), (\mathsf{pk}_i, \hat{u}_i, 0) : i \in [N]\}$,
   - $\mathcal{L}_2 = \{(I_i', \mathbf{0}, r_i' = \mathtt{DLog}(I_i')) : i \in [P_1]\}$, and

- $\mathcal{L}_3 = \{(\tau(t_j), \mathbf{0}, t_j) : j \in [P_2]\}$.

(Note that $|\mathcal{L}| \le |\mathcal{L}_1| + |\mathcal{L}_2| + |\mathcal{L}_3| \le 1 + N + P_1 + P_2 = N + P + 1$.)

Online Phase:
The algorithm is given $p, \mathfrak{g} = \tau(1), \mathsf{pk}_i = \tau(x_i), 1 \le i \le N$ as input.

4. Initialize the set $\mathsf{H}_{\mathsf{resp}} = \{\}$ which stores the random oracle input/output pairs observed during online processing. It also maintains the set $\mathsf{H}_{\mathsf{Fixed}} = \{(h_1, \mathsf{H}(h_1)), \ldots, (h_{P_1}, \mathsf{H}(h_{P_1}))\}$ that $\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{Sig.pre}}$ has already fixed in Step 1.
5. $\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{bridge.on}}$ runs $\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{Sig.on}}$ with a number of access to the generic group oracles $\mathtt{GO} = (\mathtt{Mult}(\cdot,\cdot), \mathtt{Inv}(\cdot))$, the restricted discrete-log oracles $\mathtt{DLog}(\cdot)$, the signing oracles $\mathsf{Sign}_i(\cdot)$ for $1 \le i \le N$, and the random oracle $\mathsf{H}(\cdot)$. We maintain the invariant that every output of a generic group query during the online phase appears in the list $\mathcal{L}$ that is kept in track on the challenger's side. We consider the following cases:
   (a) Whenever $\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{Sig.on}}$ submits a query w to the random oracle H:
   - If there is a pair $(w, R) \in \mathsf{H}_{\mathsf{resp}}$ for some string $R \in \{0,1\}^{\lambda_1}$ then return R.
   - Otherwise, check if $(w, \mathsf{H}(w)) \in \mathsf{H}_{\mathsf{Fixed}}$. If so, then return $\mathsf{H}(w)$.
   - Otherwise, select $R \leftarrow\!\!\$ \{0,1\}^{\lambda_1}$ and add (w, R) to $\mathsf{H}_{\mathsf{resp}}$.
   - If w has the form $w = (\mathfrak{a} \| m_i)$ where the value $\mathfrak{a}$ has not been observed previously (i.e., is not in $\mathcal{L}$) then we query $b = \mathtt{DLog}(\mathfrak{a})$ which results in the challenger adding $(\mathfrak{a}, \mathbf{0}, b)$ to $\mathcal{L}$.

   (b) Whenever $\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{Sig.on}}$ submits a query $\mathfrak{a}$ to the generic group oracle $\mathtt{Inv}(\cdot)$:
   - If $\mathfrak{a}$ is not in $\mathcal{L}$ then we immediately query $b = \mathtt{DLog}(\mathfrak{a})$ which results in the challenger adding $(\mathfrak{a}, \mathbf{0}, b)$ to $\mathcal{L}$.
   - Otherwise, $(\mathfrak{a}, \mathbf{a}, b) \in \mathcal{L}$ for some $\mathbf{a}$ and b. Then we query $\mathtt{Inv}(\mathfrak{a}) = \tau(-\mathbf{a} \cdot \mathbf{x} - b)$, output the result, which results in the challenger adding $(\tau(-\mathbf{a} \cdot \mathbf{x} - b), -\mathbf{a}, -b)$ to $\mathcal{L}$.

   (c) Whenever $\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{Sig.on}}$ submits a query $\mathfrak{a}, \mathfrak{b}$ to the generic group oracle $\mathtt{Mult}(\cdot,\cdot)$:
   - If the element $\mathfrak{a}$ (resp. $\mathfrak{b}$) is not in $\mathcal{L}$ then query $b_0 = \mathtt{DLog}(\mathfrak{a})$ (resp. $b_1 = \mathtt{DLog}(\mathfrak{b})$) which results in the challenger adding the element $(\mathfrak{a}, \mathbf{0}, b_0)$ (resp. $(\mathfrak{b}, \mathbf{0}, b_1)$) to $\mathcal{L}$.
   - Otherwise both elements $(\mathfrak{a}, \mathbf{a}_0, b_0), (\mathfrak{b}, \mathbf{a}_1, b_1) \in \mathcal{L}$. Then we return $\mathtt{Mult}(\mathfrak{a}, \mathfrak{b}) = \tau((\mathbf{a}_0 + \mathbf{a}_1) \cdot \mathbf{x} + b_0 + b_1)$ which results in the challenger adding $(\tau((\mathbf{a}_0 + \mathbf{a}_1) \cdot \mathbf{x} + b_0 + b_1), \mathbf{a}_0 + \mathbf{a}_1, b_0 + b_1) \in \mathcal{L}$.

   (d) Whenever $\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{Sig.on}}$ submits a query m_i to the signing oracle $\mathsf{Sign}(x_j, \cdot)$:
   - The attacker tries to forge a signature without knowledge of the secret key x_j, relying on the ability to program the random oracle as follows:
     (i) Pick s_i, e_i randomly and compute $I_i = \tau(s_i - x_j e_i) = \mathtt{Mult}(\mathfrak{s}_i, \mathtt{Pow}(\mathsf{pk}_j, -e_i))$ where $\mathfrak{s}_i = \mathtt{Pow}(\mathfrak{g}, s_i)$.
     (ii) If $\mathsf{H}(I_i \| m_i)$ has been previously queried or if it is in the set $\mathsf{H}_{\mathsf{Fixed}}$, then return $\perp$.
     (iii) Otherwise, program $\mathsf{H}(I_i \| m_i) := e_i$ and return $\sigma_i = (s_i, e_i)$.
   - We remark that a side effect of querying the Sign_j oracle is the addition of the tuples $(\tau(s_i), \mathbf{0}, s_i)$, $(\tau(x_j e_i), e_i \hat{u}_i, 0)$ and $(\tau(s_i - x_j e_i), -e_i \hat{u}_i, s_i)$ to $\mathcal{L}$, since these values are computed using the generic group oracles $\mathtt{Inv}, \mathtt{Mult}$.
6. After $\mathcal{A}^{\mathsf{on}}_{\mathsf{sig}}$ outputs $\sigma_{i*} = (s_{i*}, e_{i*})$ and m_{i*}, identify the index $i* \in [N]$ such that $\mathsf{Vf}(\mathsf{pk}_{i*}, m_{i*}, \sigma_{i*}) = 1$. Without loss of generality, we can assume that m_{i*} is not a part of pre-fixed points $\mathsf{H}_{\mathsf{Fixed}}$ in Step 2 since if the attacker forges a signature (s_{i*}, e_{i*}) for a message m_{i*} which involves the pre-fixed point $(I_{i*} \| m_{i*}, \mathsf{H}(I_{i*} \| m_{i*})) \in \mathsf{H}_{\mathsf{Fixed}}$ then we can directly force the bridge event to occur, i.e., $(I_{i*}, \mathbf{0}, r_{i*} = \mathtt{DLog}(I_{i*})) \in \mathcal{L}$ and $(I_{i*}, -e_{i*}\hat{u}_{i*}, s_{i*}) \in \mathcal{L}$ (see Step 7 below) where $\mathbf{0} \ne -e_{i*}\hat{u}_{i*}$.
7. Compute $\tau(-e_{i*}x_{i*}) = \mathtt{Inv}(\mathtt{Pow}(\tau(x_{i*}), e_{i*}))$. This will ensure that the elements $(\tau(-e_{i*}x_{i*}), -e_{i*}\hat{u}_{i*}, 0)$ and $(\tau(e_{i*}x_{i*}), e_{i*}\hat{u}_{i*}, 0)$ are both added to $\mathcal{L}$.
8. Compute $\mathfrak{s}_{i*} = \mathtt{Pow}(\mathfrak{g}, s_{i*})$ to ensure that $(\mathfrak{s}_{i*}, \mathbf{0}, s_{i*}) \in \mathcal{L}$.
9. Finally, compute $I_{i*} = \mathtt{Mult}(\mathfrak{s}_{i*}, \tau(-e_{i*}x_{i*})) = \tau(s_{i*} - x_{i*}e_{i*})$ which ensures that $(I_{i*}, -e_{i*}\hat{u}_{i*}, s_{i*}) \in \mathcal{L}$ and check to see if we previously had any tuple of the form $(I_{i*}, \mathbf{a}, b) \in \mathcal{L}$.

Analysis. We first remark that if the signature is valid then we must have $e_{i*} = \mathsf{H}(I_{i*} \| m_{i*})$ and $\mathtt{DLog}(I_{i*}) = s_{i*} - x_{i*}e_{i*} = \mathbf{a} \cdot \mathbf{x} + b$. Moreover, without loss of generality, we can assume that each string $\mathfrak{y}$ occurs at most once in the list $\mathcal{L}$ in Step 5 because if at any point we have some string $\mathfrak{y}$ such that $(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L}$ and $(\mathfrak{y}, \mathbf{c}, d) \in \mathcal{L}$ for $(\mathbf{a}, b) \ne (\mathbf{c}, d)$ then we can immediately have a BRIDGE^N instance $(\tau((\mathbf{a} - \mathbf{c}) \cdot \mathbf{x}), \mathbf{a} - \mathbf{c}, 0) \in \mathcal{L}$ and $(\tau(d - b), \mathbf{0}, d - b) \in \mathcal{L}$ since $\tau((\mathbf{a} - \mathbf{c}) \cdot \mathbf{x}) = \tau(d - b)$.

We now define the failure events $\mathsf{FailtoSign}$, $\mathsf{FailtoFind}(I_{i*})$, and $\mathsf{BadQuery}$. The event $\mathsf{FailtoSign}$ occurs when our reduction outputs $\perp$ in Step 5.(d) due to the signing oracle failure, i.e., since $\mathsf{H}(I_i \| m_i)$ has been previously queried or it is contained in the set $\mathsf{H}_{\mathsf{Fixed}}$. $\mathsf{FailtoFind}(I_{i*})$ denotes the event that we find that the signature is valid but I_{i*} was not previously recorded in our list $\mathcal{L}$ before we computed $\mathtt{Mult}(\mathfrak{s}_{i*}, \tau(-e_{i*}x_{i*}))$ in the last step so that we cannot find the bridge event. $\mathsf{BadQuery}$ is the event that the signature is valid but for the only prior tuple $(I_{i*}, \mathbf{a}, b) \in \mathcal{L}$ we have that $\mathbf{a} = -e_{i*}\hat{u}_{i*}$ so that we still cannot find the bridge event.

Claim 7, Claim 8, and Claim 9 upper bound the probability of our events $\mathsf{FailtoSign}$, $\mathsf{FailtoFind}$, and $\mathsf{BadQuery}$ respectively.

Claim 7. $\Pr[\mathsf{FailtoSign}] \le \dfrac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)}{p}$.

Proof of Claim 7. We observe that every time the attacker queries the signing oracle, we would generate a query to the random oracle H. Thus, we would have at most $q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1$ input/output pairs recorded for the random oracle, including the fixed points during the preprocessing phase. Since $I_i = \tau(s_i - x_i e_i)$ represents a fresh/randomly selected group element of size p, the probability that (I_i, m_i) is one of the inputs is at most $(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)/p$. Applying union bound over $q_{\mathsf{S}}^{\mathsf{on}}$ queries to the signing oracle, we have that

$$\Pr[\mathsf{FailtoSign}] \le \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)}{p}. \qquad \square$$

Claim 8. $\Pr[\mathsf{FailtoFind}(I_{i*})] \le \dfrac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1}{p - |\mathcal{L}|} + \dfrac{1}{2^{\lambda_1}}$.

Proof of Claim 8. We observe that if $I_{i*} \notin \mathcal{L}$ then we can view $I_{i*} = \tau(s_{i*} - x_{i*}e_{i*})$ as a uniformly random binary string from a set of size at least $p - |\mathcal{L}|$ which had not yet been selected at the time $\mathcal{A}_{\mathsf{Sig}}^{\mathsf{on}}$ outputs σ_{i*}. Thus, the probability that the query $\mathsf{H}(I_{i*} \| m_{i*})$ was previously recorded is at most $(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)/(p - |\mathcal{L}|)$. If the query $\mathsf{H}(I_{i*} \| m_{i*})$ was *not* previously recorded then the probability of a successful forgery $\mathsf{H}(I_{i*} \| m_{i*}) = e_{i*}$ is *at most* 2^{-k_1} since we can view $\mathsf{H}(I_{i*} \| m_{i*})$ chosen uniformly at random from the possible 2^{λ_1} options. Hence, we have that

$$\Pr[\mathsf{FailtoFind}(I_{i*})] \le \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1}{p - |\mathcal{L}|} + \frac{1}{2^{\lambda_1}}. \qquad \square$$

Claim 9. $\Pr[\mathsf{BadQuery}] \le \dfrac{q_{\mathsf{H}}^{\mathsf{on}}}{2^{\lambda_1}}$.

Proof of Claim 9. We observe that by construction we ensure that the tuple $(I, \mathbf{a}, b)$ will always be recorded in $\mathcal{L}$ before a query of the form $\mathsf{H}(I \| m)$ is ever issued — if I is new then we call $\mathtt{DLog}(I)$ before querying the random oracle. Now define a subset $\widehat{\mathcal{L}} \subset \mathcal{L}$ as the set of tuples $(\mathfrak{a}, \mathbf{a}, b) \in \mathbb{G} \times \mathbb{Z}_p^N \times \mathbb{Z}_p$ such that $\mathbf{a}$ has exactly *one* nonzero element. Now we call a random oracle query $x = (I \| m)$ "bad" if $\mathsf{H}(x) = -\widetilde{a}$ where the tuple $(I, \mathbf{a}, b) \in \widehat{\mathcal{L}}$ has already been recorded and the nonzero element of $\mathbf{a}$ is $\widetilde{a}$ (Recall that if there were two recorded tuples $(I, \mathbf{a}, b)$ and $(I, \mathbf{c}, d)$ then our algorithm would have already found a BRIDGE^N instance). Thus, the probability each individual query is "bad" is at most $2/2^{\lambda_1}$ and we can use union bounds to upper bound the probability of any "bad" query as

$$\Pr[\mathsf{BadQuery}] \le \frac{q_{\mathsf{H}}^{\mathsf{on}}}{2^{\lambda_1}}. \qquad \square$$

Now we have shown that

$$\begin{aligned}
&\Pr\left[\mathsf{BridgeChal}_{\mathcal{A}_{\mathsf{bridge}}}^{\tau,N}(\lambda) = 1\right] \\
&\ge \Pr\left[\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A}^{\mathsf{BF\text{-}RO}}_{\mathsf{Sig.on,str}_{\tau,\mathsf{H}}},\Pi^{\mathsf{short}}_{\setminus\{0\}}}(\lambda) = 1\right] - \Pr[\mathsf{FailtoSign}] \\
&\quad - \Pr[\mathsf{FailtoFind}(I_{i*})] - \Pr[\mathsf{BadQuery}] \\
&\ge \Pr\left[\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A}^{\mathsf{BF\text{-}RO}}_{\mathsf{Sig.on,str}_{\tau,\mathsf{H}}},\Pi^{\mathsf{short}}_{\setminus\{0\}}}(\lambda) = 1\right] - \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)}{p} \\
&\quad - \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1}{p - |\mathcal{L}|} - \frac{q_{\mathsf{H}}^{\mathsf{on}} + 1}{2^{\lambda_1}}.
\end{aligned}$$

Finally, by applying Lemma 4.3, we can conclude that

$$\begin{aligned}
&\Pr\left[\mathsf{SigForge}^{\tau,\mathsf{H},N}_{\mathcal{A}^{\mathsf{BF\text{-}RO}}_{\mathsf{Sig.on,str}_{\tau,\mathsf{H}}},\Pi^{\mathsf{short}}_{\setminus\{0\}}}(\lambda) = 1\right] \\
&\le \left[\mathsf{BridgeChal}_{\mathcal{A}_{\mathsf{bridge}}}^{\tau,N}(\lambda) = 1\right] + \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1)}{p} \\
&\quad + \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P_1}{p - |\mathcal{L}|} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + 1}{2^{\lambda_1}} \\
&\le \frac{q_{\mathsf{G}}^{\mathsf{on}}(N+P) + 3q_{\mathsf{G}}^{\mathsf{on}}(q_{\mathsf{G}}^{\mathsf{on}} + 1)/2}{p - 2P(N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1) - (N + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)^2 - N} + \frac{q_{\mathsf{S}}^{\mathsf{on}}(q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P)}{p} \\
&\quad + \frac{q_{\mathsf{H}}^{\mathsf{on}} + q_{\mathsf{S}}^{\mathsf{on}} + P}{p - (N + P + 3q_{\mathsf{G}}^{\mathsf{on}} + 1)} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + 1}{2^{\lambda_1}}. \qquad \square
\end{aligned}$$

Reminder of Lemma 5.2. *Let $p > 2^{2\lambda}$ is a prime number. Let $\mathcal{A} := \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}\right)$ be a pair of bit-fixing generic algorithms with a labeling map $\tau : \mathbb{Z}_p \to \mathbb{G}$ such that $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}GG}(P)}$ fixes P input/output pairs of the labeling map τ and $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$ makes at most $q_{\mathsf{G}}^{\mathsf{on}} := q_{\mathsf{G}}^{\mathsf{on}}(\lambda)$ queries to the generic group oracles and $q_{\mathcal{O}}^{\mathsf{on}} := q_{\mathcal{O}}^{\mathsf{on}}(\lambda)$ queries to the oracle $\mathcal{O}$. Then $\Pr\left[\mathsf{QDBridgeChal}_{\mathcal{A}}^{\tau}(\lambda) = 1\right] \le \varepsilon$, where*

$$\varepsilon := \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathcal{O}}^{\mathsf{on}})^2 + (5 + 2P)(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathcal{O}}^{\mathsf{on}}) + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathcal{O}}^{\mathsf{on}} + 1) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathcal{O}}^{\mathsf{on}} + 1)^2 - 2(2q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathcal{O}}^{\mathsf{on}})},$$

in the GGM of prime order p, where the randomness is taken over the selection of $x_1, \ldots, x_N, \tau$ as well as any random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}GG}(P)}$.

Proof of Lemma 5.2. The proof works by maintaining a list $\mathcal{L}$ of tuples $(\tau(y), \mathbf{a}, b)$ where $\tau(y) \in \mathbb{G}$, $\mathbf{a} \in \mathbb{Z}_p^{\dim(\mathbf{x})}$, and $b \in \mathbb{Z}_p$ such that $y = \mathbf{a} \cdot \mathbf{x} + b$ for every oracle output $\tau(y)$. To bound the probability that the attacker wins the quadratic bridge-finding game, we consider the case where the event $\mathsf{QDBridge}$ occurs after the attacker makes q_{G} generic group queries and $q_{\mathcal{O}}$ queries to $\mathcal{O}$. Suppose that $\dim(\mathbf{x}) = j$ at the point in time when $\mathsf{QDBridge}$ occurs. We observe that $j \le 2q_{\mathsf{G}} + q_{\mathcal{O}} + 1$ since $\dim(\mathbf{x})$ can be increased by at most 2 whenever $\mathcal{A}$ makes a generic group query and at most 1 whenever $\mathcal{A}$ makes a query to $\mathcal{O}$. We further observe that in this case, $|\mathcal{L}| \le 3q_{\mathsf{G}} + q_{\mathcal{O}} + 1$ since for each query to $\mathcal{O}$, at most 1 item can be added to $\mathcal{L}$ whereas each generic group oracle query can add at most 3 times to $\mathcal{L}$ (exactly three in the case when $\mathcal{A}$ queries $\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2)$ on two fresh elements).

Now consider the event BRIDGE where the list $\mathcal{L}$ contains two distinct tuples $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$ such that $\mathbf{a} \cdot \mathbf{x} + b = \mathbf{a}' \cdot \mathbf{x} + b'$. To upper bound $\Pr[\mathsf{BRIDGE}]$, consider the event $\overline{\mathsf{BRIDGE}_{<i}}$ that the event BRIDGE has not occurred until the $(i-1)^{th}$ query. Conditioning on the event $\overline{\mathsf{BRIDGE}_{<i}}$, we are now interested in the event BRIDGE_i where the i^{th} query makes the event BRIDGE occur, i.e., a tuple $(\mathfrak{y}_i, \mathbf{a}_i, b_i)$ has been recorded to $\mathcal{L}$ and there exists a tuple of the form $(\cdot, \mathbf{a}, b)$ such that $\mathbf{a}_i \cdot \mathbf{x} + b_i = \mathbf{a} \cdot \mathbf{x} + b$. We can

essentially view $\mathbf{x}$ sampled uniformly at random subject to some restrictions due to $\overline{\mathsf{BRIDGE}_{<i}}$, from which we know that for any distinct pair $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$ we have $\mathbf{a} \cdot \mathbf{x} + b \neq \mathbf{a}' \cdot \mathbf{x} + b'$.

Let $r \leq \dim(\mathbf{x}) = j$ be an index such that $\mathbf{a}[r] - \mathbf{a}'[r] \neq 0$ and suppose that $x_r = \mathbf{x}[r]$ is the last value sampled. We also define two subsets $\mathcal{L}_0, \mathcal{L}_1 \subseteq \mathcal{L}$ where $\mathcal{L}_0 \coloneqq \{(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L} : \mathbf{a} = \mathbf{0}\}$ and $\mathcal{L}_1 \coloneqq \{(\mathfrak{y}, \mathbf{a}, b) \in \mathcal{L} : \mathbf{a} \neq \mathbf{0}\}$. Now we observe that $\mathcal{L}_0 \cup \mathcal{L}_1 = \mathcal{L}$ and if we pick two distinct pairs $(\mathfrak{y}_1, \mathbf{a}_1, b_1)$ and $(\mathfrak{y}_2, \mathbf{a}_2, b_2)$ both from $\mathcal{L}_0$ we are guaranteed to have that $\mathbf{a}_1 \cdot \mathbf{x} + b_1 \neq \mathbf{a}_2 \cdot \mathbf{x} + b_2$ since $\mathbf{a}_1 = \mathbf{a}_2 = \mathbf{0}$, which implies that we are guaranteed to not cause any bridge event. At this point, we can view x_r as being drawn uniformly at random from a set of at least $p - (|\mathcal{L}|^2 - |\mathcal{L}_0|^2) - (j-1)$ remaining values, subject to all of the restrictions. To see this, we observe the following:

- x_r must be distinct from $x_1, \ldots, x_{r-1}$ since otherwise we would have had a bridge event, and
- each pair of distinct elements $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}$, we have the following cases:
  - if $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_1$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_1$, then this pair eliminates at most one possible value of x_r, i.e., since $\mathfrak{y}_j$ and $\mathfrak{y}_k$ are distinct, we have $\mathbf{a}_j \cdot \mathbf{x} + b_j \neq \mathbf{a}_k \cdot \mathbf{x} + b_k$, which implies that $x_r \neq \left((b_k - b_j) - \sum_{i=1, i\neq j}^{N} (\mathbf{a}_k[i] - \mathbf{a}_j[i]) x_i\right) (\mathbf{a}_j[r] - \mathbf{a}_k[r])^{-1}$,
  - if $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_1$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_0$, then this pair eliminates at most one possible value of x_r, i.e., since $\mathfrak{y}_j$ and $\mathfrak{y}_k$ are distinct, we have $\mathbf{a}_j \cdot \mathbf{x} + b_j \neq \mathbf{0} \cdot \mathbf{x} + b_k$, which implies that $x_r \neq \left((b_k - b_j) + \sum_{i=1, i\neq j}^{N} \mathbf{a}_j[i] x_i\right) \mathbf{a}_j[r]^{-1}$,
  - $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_0$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_1$, then this pair eliminates at most one possible value of x_r, i.e., since $\mathfrak{y}_j$ and $\mathfrak{y}_k$ are distinct, we have $\mathbf{0} \cdot \mathbf{x} + b_j \neq \mathbf{a}_k \cdot \mathbf{x} + b_k$, which implies that $x_r \neq \left((b_k - b_j) - \sum_{i=1, i\neq j}^{N} \mathbf{a}_k[i] x_i\right) (-\mathbf{a}_k[r])^{-1}$, and
  - if both $(\mathfrak{y}_j, \mathbf{a}_j, b_j) \in \mathcal{L}_0$ and $(\mathfrak{y}_k, \mathbf{a}_k, b_k) \in \mathcal{L}_0$, then since both $\mathbf{a}_j = \mathbf{a}_k = \mathbf{0}$, it is a contradiction that $\mathbf{a}_j[r] - \mathbf{a}_k[r] \neq 0$ and therefore we do not eliminate any value of x_r.
- Hence, the total number of values that we remove when we draw x_r is at most $(r-1) + |\mathcal{L}_1|^2 + 2|\mathcal{L}_0||\mathcal{L}_1| \leq (j-1) + (|\mathcal{L}|^2 - |\mathcal{L}_0|^2)$.

We also observe that $|\mathcal{L}| \leq P + 3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1$ and $|\mathcal{L}_0| \geq P$ since the list already contains P additional pre-fixed pairs and each generic group oracle query adds *at most* three new tuples to $\mathcal{L}$ — exactly three in the case that we query $\mathtt{Mult}(\mathfrak{y}_1, \mathfrak{y}_2)$ on two fresh elements.

Thus, the probability that $\mathbf{a}_i \cdot \mathbf{x} + b_i = \mathbf{a} \cdot \mathbf{x} + b$ is at most

$$\frac{1}{p - (P + 3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 + P^2 - (2q_{\mathsf{G}}^{\mathsf{on}} + 2q_{O}^{\mathsf{on}})}.$$

Union bounding over all tuples in $\mathcal{L}$, we have

$$\begin{aligned}
&\Pr[\mathsf{BRIDGE}] \\
&= \sum_{i \leq q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}}} \Pr\left[\mathsf{BRIDGE}_i \middle| \overline{\mathsf{BRIDGE}_{<i}}\right] \\
&\leq \sum_{i \leq q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}}} \frac{1 + P + 3i}{p - (P + 3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 + P^2 - (2q_{\mathsf{G}}^{\mathsf{on}} + 2q_{O}^{\mathsf{on}})} \\
&\leq \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})^2 + (5 + 2P)(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 - 2(2q_{\mathsf{G}}^{\mathsf{on}} + 2q_{O}^{\mathsf{on}})}.
\end{aligned}$$

Now, consider the case where the event BRIDGE never occurs. Then we would like to upper bound the probability

$$\Pr\left[\mathsf{QDBridgeChal}_{\mathcal{A}}^{\tau,O}(\lambda) = 1 \middle| \overline{\mathsf{BRIDGE}}\right].$$

The argument works similarly; conditioning on the event $\overline{\mathsf{BRIDGE}}$, we know that for any distinct pair $(\mathfrak{y}, \mathbf{a}, b)$ and $(\mathfrak{y}', \mathbf{a}', b')$ we have $\mathbf{a} \cdot \mathbf{x} + b \neq \mathbf{a}' \cdot \mathbf{x} + b'$. Let $r \leq \dim(\mathbf{x}) \leq 2q_{\mathsf{G}} + q_{O} + 1$ be an index such that $\mathbf{a}[r] - \mathbf{a}'[r] \neq 0$ and suppose that $x_r = \mathbf{x}[r]$ is the last value sampled. At this point, we can view x_r as being drawn uniformly at random from a set of at least $p - (|\mathcal{L}|^2 - |\mathcal{L}_0|^2) - (2q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})$ remaining values as before. Now, the attacker wins if and only if s/he outputs a tuple $(i_1, i_2, \mathbf{a}, b)$ such that $x_{i_1} x_{i_2} = \mathbf{a} \cdot \mathbf{x} + b$. Since this equation is at most degree 2 in terms of x_r (in case $i_1 = i_2 = r$), there are at most 2 roots in $\mathbb{Z}_p$. Thus, we have that

$$\begin{aligned}
&\Pr\left[\mathsf{QDBridgeChal}_{\mathcal{A}}^{\tau,O}(\lambda) = 1 \middle| \overline{\mathsf{BRIDGE}}\right] \\
&\leq \frac{2}{p - (|\mathcal{L}|^2 - |\mathcal{L}_0|^2) - (2q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})} \\
&\leq \frac{2}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 - (2q_{\mathsf{G}}^{\mathsf{on}} + 2q_{O}^{\mathsf{on}})}.
\end{aligned}$$

Taken together, we have

$$\begin{aligned}
&\Pr\left[\mathsf{QDBridgeChal}_{\mathcal{A}}^{\tau,O}(\lambda) = 1\right] \\
&\leq \Pr[\mathsf{BRIDGE}] + \Pr\left[\mathsf{QDBridgeChal}_{\mathcal{A}}^{\tau,O}(\lambda) = 1 \middle| \overline{\mathsf{BRIDGE}}\right] \\
&\leq \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})^2 + (5 + 2P)(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 - 2(2q_{\mathsf{G}}^{\mathsf{on}} + 2q_{O}^{\mathsf{on}})} \\
&\quad + \frac{2}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 - (2q_{\mathsf{G}}^{\mathsf{on}} + 2q_{O}^{\mathsf{on}})} \\
&= \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})^2 + (5 + 2P)(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}}) + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 - 2(2q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})},
\end{aligned}$$

which completes the proof. □

Reminder of Theorem 5.3. *Let* $\Pi = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *be PSEC-KEM and* $p > 2^{2\lambda}$ *be a prime number. Let* $\mathcal{A} = \left(\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}, \mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}\right)$ *be a pair of bit-fixing generic algorithms with a labeling map* $\tau : \mathbb{Z}_p \to \mathbb{G}$ *such that* $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ *fixes* $P_{1,1}$ *(resp.* $P_{1,2}, P_{1,3}$*) input/output pairs of a random oracle* $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_p$ *(resp.* $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{\lambda}$, $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{\lambda_1}$*), and* P_2 *input/output pairs of the map* τ *such that* $P_{1,1} + P_{1,2} + P_{1,3} = P_1$ *and* $2P_1 + P_2 = P$*, and the hint* $\mathsf{str}_{\tau,\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2}$ *is only dependent on those* P *points. If* $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ *makes at most* $q_{\mathsf{H}}^{\mathsf{on}}$ *(resp.* $q_{\mathsf{G}}^{\mathsf{on}}, q_{\mathsf{E}}^{\mathsf{on}}$*) queries to the random oracle (resp. generic group oracle, encapsulation oracle), then* $\Pr\left[\mathsf{KEM}^{\tau,(\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2),\mathsf{cpa}}_{\mathcal{A}_{\mathsf{on},\mathsf{str}_{\tau,\mathsf{H}_0,\mathsf{H}_1,\mathsf{H}_2}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \varepsilon$*, with*

$$\varepsilon = \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}})^2 + (5+2P)(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}}) + 4}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1)^2 - 2(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{E}}^{\mathsf{on}})} + \frac{3(q_{\mathsf{H}}^{\mathsf{on}} + P)q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}} + \frac{(q_{\mathsf{H}}^{\mathsf{on}} + P)q_{\mathsf{E}}^{\mathsf{on}}}{p},$$

where the randomness is taken over the selection of τ and the random coins of $\mathcal{A}_{\mathsf{on}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$.

Proof of Theorem 5.3. We use a hybrid argument to prove the CPA security of PSEC-KEM in the BF-ROM+GGM. The hybrids work the same as the proof of Theorem D.2 during the online phase, except that the distinguisher $\mathcal{D} = (\mathcal{D}_{\mathsf{pre}}, \mathcal{D}_{\mathsf{on}})$ in the preprocessing phase gets the hint generated from $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$. In particular, $\mathcal{A}_{\mathsf{pre}}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}$ fixes

- $P_{1,1}$ input/output pairs $(\{(h_{1,1}, \mathsf{H}_0(h_{1,1})), \ldots, (h_{1,P_{1,1}}, \mathsf{H}_0(h_{1,P_{1,1}}))\})$ of H_0,
- $P_{1,2}$ input/output pairs $(\{(h_{2,1}, \mathsf{H}_1(h_{2,1})), \ldots, (h_{2,P_{1,2}}, \mathsf{H}_1(h_{2,P_{1,2}}))\})$ of H_1,
- $P_{1,3}$ input/output pairs $(\{(h_{3,1}, \mathsf{H}_0(h_{3,1})), \ldots, (h_{3,P_{1,3}}, \mathsf{H}_0(h_{3,P_{1,3}}))\})$ of H_2, and
- P_2 input/output pairs $(\{(t_1, \tau(t_1)), \ldots, (t_{P_2}, \tau(t_{P_2}))\})$ of τ.

Furthermore, for each fixed RO input/output pair $(h_j, \mathsf{H}_i(h_j))$ for $i = 0, 1, 2$, $\mathcal{D}_{\mathsf{pre}}$ first parses $h_j = I_j \| I'_j$ where I_j is a bitstring of length ℓ. If I_j is a fresh ℓ-bit string, i.e., $I_j \neq \tau(t_k)$ for all $k \in [P_2]$, then $\mathcal{D}_{\mathsf{pre}}$ picks a value for $\tau^{-1}(I_j) = r_j$ where the labeling map τ will be sampled later. Note that this is possible because the labeling map τ has not been sampled yet. Finally, $\mathcal{D}_{\mathsf{pre}}$ additionally fixes the input/output pairs of τ such as $\{(r_1, I_1), \ldots, (r_{P_1}, I_{P_1})\}$. This leads to having $P_{1,1} + P_{1,2} + P_{1,3} = P_1$ fixed RO points and $P_1 + P_2$ fixed points of τ for the distinguisher, having in total $P_1 + (P_1 + P_2) = P$ fixed points as the hint for the online phase. For completeness, we restate the hybrids below.

Hybrid H_0. This is the original CPA indistinguishability game $\mathsf{KEM}_{\mathcal{A},\Pi}^{\tau,\mathsf{H},\mathsf{cpa}}(\lambda)$ for PSEC-KEM with the challenge bit $b = 0$. In particular, $\mathcal{D}_{\mathsf{on}}$ has access to the encapsulation oracle $\mathsf{Encaps}'_0(\cdot) \coloneqq \mathsf{Encaps}(\cdot, 1^\lambda)$, i.e., whenever $\mathcal{D}_{\mathsf{on}}$ submits a query $\mathsf{Encaps}'_0(\mathsf{pk})$, $\mathcal{D}_{\mathsf{on}}$ always gets the output $\mathsf{Encaps}(\mathsf{pk}, 1^\lambda)$.

Hybrid H_1. This is the same security game with H_0 except that the encapsulation oracle $\mathsf{Encaps}'_0(\cdot)$ is replaced with a modified oracle $\mathsf{Encaps}'_1(\cdot)$. In $\mathsf{Encaps}'_1(\cdot)$, $\tau(\alpha)$ and $\tau(\alpha x)$ (where x is the secret key) are replaced with random elements in $\tau(\mathbb{Z}_p)$ by querying the oracle O twice. See Figure 2 for the details.

Hybrid H_2. This is the same security game with H_1 except that the encapsulation oracle $\mathsf{Encaps}'_1(\cdot)$ is further replaced with a modified oracle $\mathsf{Encaps}'_2(\cdot)$. $\mathsf{Encaps}'_2(\cdot)$ is exactly the same as $\mathsf{Encaps}'_1(\cdot)$ except that the key k is sampled uniformly at random from $\{0,1\}^\lambda$.

Hybrid H_3. This is the same security game with H_2 except that the encapsulation oracle $\mathsf{Encaps}'_2(\cdot)$ is replaced with a modified oracle $\mathsf{Encaps}'_3(\cdot) \coloneqq \mathsf{Encaps}_1(\cdot)$, which leads to the original CPA indistinguishability game $\mathsf{KEM}_{\mathcal{A},\Pi}^{\tau,\mathsf{H},\mathsf{cpa}}(\lambda)$ for PSEC-KEM with the challenge bit $b = 1$. In particular, $\mathsf{Encaps}'_3(\cdot)$ is the same as $\mathsf{Encaps}'_2(\cdot)$ but the random elements in $\tau(\mathbb{Z}_p)$ are reverted back to the honest computation $\tau(\alpha)$ and $\tau(\alpha x)$. This can also be interpreted as replacing an honest key k in $\mathsf{Encaps}'_0(\cdot)$ with a uniformly random key of length λ. See Figure 2 for the details.

Indistinguishability of H_0 and H_1. With a similar reasoning, we can argue that two hybrids are perfectly indistinguishable unless one of the three events occurs: for some round $i \leq q_{\mathsf{E}}^{\mathsf{on}}$,

(1) the random oracle query $\mathsf{H}_2(\tau(\alpha_i)\|\tau(\alpha_i x))$ has been made,
(2) the adversary makes query r_i to H_0 (which is not one of the fixed points of H_0) but has not queried $(\tau(\alpha_i)\|\tau(\alpha_i x))$ to H_2, or
(3) r_i is one of the fixed points of H_0.

We can argue in the same way as Theorem D.2 that Case (1) leads to winning the quadratic bridge-finding game with preprocessing.

Let Lucky_1 be the event that Case (2) occurs. If r is not one of the fixed points of H_0, then it is essentially the same as Case (2) in the proof of Theorem D.2. Hence,

$$\Pr[\mathsf{Lucky}_1] \leq \frac{q_{\mathsf{H}}^{\mathsf{on}} \cdot q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}}.$$

Let Lucky_2 be the event that Case (3) occurs. Since there are at most P fixed points, union bounding over $q_{\mathsf{E}}^{\mathsf{on}}$ queries, we have

$$\Pr[\mathsf{Lucky}_2] \leq \frac{P \cdot q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}}.$$

Taken all together with Lemma 5.2, we have

$$\begin{aligned}
&\left|\Pr[\mathcal{D}^{H_0} = 1] - \Pr[\mathcal{D}^{H_1} = 1]\right| \\
&\leq \Pr[\mathsf{QDBridgeChal}_{\mathcal{D}}^{\tau}(\lambda) = 1] + \Pr[\mathsf{Lucky}_1] + \Pr[\mathsf{Lucky}_2] \\
&\leq \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})^2 + (5+2P)(q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}}) + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}} + 1)^2 - 2(2q_{\mathsf{G}}^{\mathsf{on}} + q_{O}^{\mathsf{on}})} \\
&\quad + \frac{q_{\mathsf{H}}^{\mathsf{on}} \cdot q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}} + \frac{P \cdot q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}} \\
&\leq \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}})^2 + (5+2P)(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}}) + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1)^2 - 4(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{E}}^{\mathsf{on}})} \\
&\quad + \frac{(q_{\mathsf{H}}^{\mathsf{on}} + P) \cdot q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}},
\end{aligned}$$

since $q_{O}^{\mathsf{on}} \leq 2q_{\mathsf{E}}^{\mathsf{on}}$.

Indistinguishability of H_1 and H_2. Again, we observe that two hybrids are perfectly indistinguishable unless the random oracle query $\mathsf{H}_1(r_i)$ is queried (which is not one of the fixed points) *or* r_i is one of the fixed points of H_1 for some round i. If r_i was not one of the fixed points of H_1, there are still several ways to retrieve r_i:

- if the random oracle query $\mathsf{H}_2(\mathfrak{y}_{1,i}\|\mathfrak{y}_{2,i})$ was made that is not one of the fixed points of H_2, or
- $\mathfrak{y}_{1,i}\|\mathfrak{y}_{2,i}$ is one of the fixed points of H_2.

In either case, one can retrieve $r_i \leftarrow c_{1_i} \oplus \mathsf{H}_2(\mathfrak{y}_{1,i}\|\mathfrak{y}_{2,i})$. Note that since the attacker only gets $(k_i, c_i = (\mathfrak{y}_{1,i}, c_{1,i}))$ as the output, s/he can first check if $\mathfrak{y}_{1,i}$ is the prefix of one of the fixed points of H_2 (let's say this fixed point is $\widetilde{\mathfrak{y}}$) and check if $\mathsf{H}_1(c_{1,i} \oplus \mathsf{H}_2(\widetilde{\mathfrak{y}})) \overset{?}{=} k_i$. If it matches then the attacker would have known that the pair (k_i, c_i) is from Encaps'_1. Since there are at most P fixed points, union bounding over $q_{\mathsf{E}}^{\mathsf{on}}$ queries we have

$$\left|\Pr[\mathcal{D}^{H_1} = 1] - \Pr[\mathcal{D}^{H_2} = 1]\right| \le q_{\mathsf{E}}^{\mathsf{on}} \left(\frac{q_{\mathsf{H}}^{\mathsf{on}} + P}{p} + \frac{q_{\mathsf{H}}^{\mathsf{on}} + P}{2^{\lambda_1}} \right).$$

Indistinguishability of H_2 and H_3. As we discussed in the proof of Theorem D.2, the distinguishing probability is the same as the distinguishing probability between H_0 and H_1, which is

$$\begin{aligned}
&\left|\Pr[\mathcal{D}^{H_2} = 1] - \Pr[\mathcal{D}^{H_3} = 1]\right| \\
&\le \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}})^2 + (5+2P)(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}}) + 4}{2p - 4P(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1) - 2(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1)^2 - 4(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{E}}^{\mathsf{on}})} \\
&\quad + \frac{(q_{\mathsf{H}}^{\mathsf{on}} + P)\cdot q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}}.
\end{aligned}$$

Putting all together, we have

$$\begin{aligned}
&\Pr\left[\mathsf{KEM}^{\tau,\mathsf{H}_0,\mathsf{H}_1,\mathsf{cpa}}_{\mathcal{A}^{\mathsf{BF\text{-}RO+GG}(P_1,P_2)}_{\mathsf{on},\mathsf{str}_{\tau,\mathsf{H}_0,\mathsf{H}_1}},\Pi}(\lambda) = 1\right] \\
&\le \frac{1}{2} + \left|\Pr[\mathcal{D}^{H_0} = 1] - \Pr[\mathcal{D}^{H_3} = 1]\right| \\
&\le \frac{1}{2} + \sum_{i=0}^{2} \left|\Pr[\mathcal{D}^{H_i} = 1] - \Pr[\mathcal{D}^{H_{i+1}} = 1]\right| \\
&\le \frac{3(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}})^2 + (5+2P)(q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}}) + 4}{p - 2P(3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1) - (3q_{\mathsf{G}}^{\mathsf{on}} + 2q_{\mathsf{E}}^{\mathsf{on}} + 1)^2 - 4(q_{\mathsf{G}}^{\mathsf{on}} + q_{\mathsf{E}}^{\mathsf{on}})} \\
&\quad + \frac{3(q_{\mathsf{H}}^{\mathsf{on}} + P)q_{\mathsf{E}}^{\mathsf{on}}}{2^{\lambda_1}} + \frac{(q_{\mathsf{H}}^{\mathsf{on}} + P)q_{\mathsf{E}}^{\mathsf{on}}}{p} + \frac{1}{2}. \qquad \square
\end{aligned}$$

H Missing Tables

S	2^{64}			2^{72}			2^{80}		
q	2^{80}	2^{85}	2^{90}	2^{80}	2^{85}	2^{90}	2^{80}	2^{85}	2^{90}
	Attacker Success Probability (Approximate)								
$p = 2^{256}$	2^{-31}	2^{-21}	2^{-11}	2^{-23}	2^{-13}	2^{-3}	2^{-15}	2^{-5}	≈ 1
$p = 2^{384}$	2^{-159}	2^{-149}	2^{-139}	2^{-151}	2^{-141}	2^{-131}	2^{-143}	2^{-133}	2^{-123}
$p = 2^{448}$	2^{-223}	2^{-213}	2^{-203}	2^{-215}	2^{-205}	2^{-195}	2^{-207}	2^{-197}	2^{-187}
$p = 2^{512}$	2^{-287}	2^{-277}	2^{-267}	2^{-279}	2^{-269}	2^{-259}	2^{-271}	2^{-261}	2^{-251}

Table 2: Security bounds of a nonzero Schnorr signature for different values of S (size of the hint), q (total query complexity), and p (group size).

Table 2 highlights that to break 128-bit security for nonzero Schnorr signatures, one would need to consider preprocessing attacks with the hint size unreasonably large which grows to the yottabyte scale ($\approx 10^{24} \approx 2^{80}$) and making 2^{90} online queries with massive computational power. However, we could even prevent this vulnerability by using larger group sizes (e.g., $p \ge 2^{384}$) which ensures that even nation-state-level attackers with significant computational resources cannot compromise nonzero Schnorr signatures except for negligibly small probability, even considering preprocessing.