Fine-Grained Non-Interactive Key-Exchange: Constructions and Lower Bounds

Abtin Afshar, Geoffroy Couteau, Mohammad Mahmoody, Elahe Sadeghi

Fine-Grained Non-Interactive Key-Exchange: Constructions and Lower Bounds

Abtin Afshar[1], Geoffroy Couteau[2] *, Mohammad Mahmoody[3] **, and Elahe Sadeghi[4]

[1] University of Virginia
abtin@virginia.edu
[2] CNRS, IRIF, Université Paris Cité, France
couteau@irif.fr
[3] University of Virginia
mohammad@virginia.edu
[4] University of Texas at Austin
elahesadeghi@utexas.edu

Abstract. In this work, we initiate a study of K-NIKE protocols in the *fine-grained* setting, in which there is a *polynomial* gap between the running time of the honest parties and that of the adversary. Our goal is to show the possibility, or impossibility, of basing such protocols on weaker assumptions than those of K-NIKE for $K \geq 3$. Our contribution is threefold.

- We show that random oracles can be used to obtain fine-grained K-NIKE protocols for *every* constant K. In particular, we show how to generalize Merkle's two-party protocol to K parties in such a way that the honest parties ask n queries each, while the adversary needs $n^{K/(K-1)}$ queries to the random oracle to find the key.
- We then improve the security by further using algebraic structures, while avoiding pairings. In particular, we show that there is a 4-party NIKE in Shoup's generic group model with a *quadratic* gap between the number of queries by the honest parties vs. that of the adversary.
- Finally, we show a limitation of using purely algebraic methods for obtaining 3-NIKE. In particular, we show that any n-query 3-NIKE protocol in Maurer's generic group model can be broken by a $O(n^2)$-query attacker. Maurer's GGM is more limited compared with Shoup's both for the parties and the adversary, as there are no explicit labels for the group elements. Despite being more limited, this model still captures the Diffie Hellman protocol. Prior to our work, it was open to break 3-NIKE protocols in Maurer's model with *any* polynomial number of queries.

* Supported by the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE), and by the France 2030 ANR Project ANR22-PECY-003 SecureCompute.
** Supported by NSF under grants CCF-1910681 and CNS1936799.

Table of Contents

1 Introduction 2
1.1 Our Contribution 3
1.2 Technical Overview: Building NIKE in the ROM and GGM 3
1.3 Technical Overview: Breaking 3-NIKE in Maurer's GGM 6
2 Preliminaries 8
3 3-NIKE in the Random Oracle Model 9
4 4-NIKE in Shoup's Generic Group Model 12
4.1 Correctness 13
4.2 Security Analysis 13
5 Impossibility Results 14
5.1 Defining 3-NIKE in Maurer's Generic Group Model 14
5.2 Breaking 3-NIKE in the MGGM without zero-test queries 16
5.3 Breaking 3-NIKE in the MGGM with zero-test queries 18

1 Introduction

Non-interactive key exchange (NIKE), introduced in the seminal work of Diffie and Hellman [DH76], is a primitive of fundamental interest in cryptography. It allows a group of parties $\mathsf{P}_1, \ldots, \mathsf{P}_k$ to simultaneously publish a single message each, such that any party can recover (without further interaction) a common group key using their secret randomness and the common messages $(\mathsf{m}_i)_{i\in[k]}$, in a way that the key remains hidden to any external observer who only gets to see $(\mathsf{m}_i)_{i\in[k]}$. NIKE is an intriguing cryptographic object: although the first construction of a 2-party NIKE was given in one of the very first papers on public key cryptography, constructing NIKE for more parties is a notoriously hard problem. Even in the two party setting, NIKE is known only from a restricted set of assumptions, such as the Diffie-Hellman assumption [DH76], the LWE assumption with super-polynomial modulus-to-noise ratio [GKRS22], and from assumptions related to the hardness of factoring [FHKP13]. In the three-party setting, constructing NIKE was a major open problem until the breakthrough result of Joux [Jou00] from the bilinear Diffie-Hellman assumption over pairing groups, which introduced what remains to date the only known construction of 3-party NIKE under a standard assumption. Furthermore, all known constructions of K-party NIKE for $K > 3$ require much heavier cryptographic machinery, such as indistinguishability obfuscation [BZ14]. Hence, as of today, K-party NIKE with $K > 3$ belongs to the world of "obfustopia" primitives (alongside with primitives such as witness encryption or functional encryption), in spite of being seemingly a much simpler primitive than obfuscation.

Fine-grained cryptography. Traditional cryptography requires hardness of cryptographic primitives to hold against arbitrary polynomial-time adversaries. In contrast, *fine-grained* cryptography aims to study the feasibility of cryptographic primitives when the adversarial power is restricted, for example, to some fixed polynomial bound. While the study of fine-grained cryptography can be traced back to the seminal paper of Merkle [Mer74, Mer78] who constructed a 2-party NIKE from idealized hash functions with security against subquadratic-time adversaries, this primitive has recently spurred a renewed interest, leading to a collection of constructions [BGI08, BHK+11, DVV16, BRSV17, BRSV18, CG18, LLW19, EWT21, DH21, WP22] and lower bounds [BM09, BC22] for fine-grained cryptographic primitives.

A core motivation underlying the research on fine-grained cryptography is the hope that by relaxing the security to hold against less powerful adversaries, it might be possible to base the existence of fine-grained primitives on assumptions which are weaker than those known to imply their full-fledged counterpart. For some types of restrictions, this has been a fruitful endeavor so far; for example, when restricting the adversary to be of *constant depth* (in the complexity classes AC^0), this has led to the construction of many standard cryptographic primitives (one-way functions, pseudorandom generators, pseudorandom functions, public key encryption), with unconditional security [DVV16]. For adversaries of logarithmic depth (in the class NC^1), this resulted in the construction of most traditional cryptographic primitives under worst-case hardness assumptions [DVV16, CG18, EWT21, WP22].

Perhaps more interestingly, some results have been achieved when restricting only the *running time* of the adversary to be bounded by some fixed polynomial in the runtime of the honest parties (the degree of the polynomial is typically called the *security gap* of the scheme). The work of [BGI08], building upon [Mer78], showed that exponentially-secure one-way functions imply *key exchange and public key encryption* with near-quadratic security gap. More recently, the work of [BC22] showed that some strong forms of average-case hardness implies *one-way functions* with near-quadratic security gap. At the other end of the hardness spectrum, the work of [BJK+18] showed the existence of "quadratically efficient" witness encryption from the LWE assumption. In each of these examples, the fine-grained primitive is built from an assumption which seems to be of a weaker nature compared to the full-fledged version.

1.1 Our Contribution

In this work, we investigate multiparty non-interactive key-exchange in the setting of fine-grained security. We focus on the setting where the adversarial runtime is restricted to be bounded by a polynomial in the honest parties' runtime. Our main motivation is to understand the possibility of basing *fine-grained* multiparty NIKE on assumptions outside of the Obfustopia realm, and ideally on some of the traditional assumptions known to imply 2-party NIKE, such as the Diffie-Hellman assumption.

Below, we always denote by n the runtime of the honest participants, and write K-NIKE for K-party NIKE. Our main results are threefold:

1. In the random oracle model, we prove the existence of a fine-grained 3-NIKE protocol with security against $o(n^{1.5})$-time adversaries. Our result generalizes to K-NIKE with security against $o(n^{1+1/(K-1)})$-time adversaries. While this result is a relatively natural generalization of the seminal protocol of Merkle, to the best of our knowledge, it has never been found before.
2. We demonstrate that larger security gaps can be achieved by additionally relying on algebraic structure: in Shoup's generic group model [Sho97], we prove the existence of a 4-NIKE with security against $o(n^2)$-time adversaries. Our result generalizes to $2K$-NIKE with security against $o(n^2)$-time adversaries in the generic $(K-1)$-linear group model. In particular, this also yields a 6-NIKE with near-quadratic hardness in the generic bilinear group model.
3. We complement our positive result by proving a limitation on the fine-grained security of K-NIKE with $K > 2$ over generic groups. In particular, we prove that for $K > 2$ any K-NIKE protocol in Maurer's generic group model [Mau05] can be broken using $O(n^2)$ queries.[5] Our result extends to the setting of K-NIKE with imperfect correctness. An important point is that, even though our impossibility result only applies to the MGGM, the Diffie-Hellman protocol for 2-NIKE *can* be stated in the MGGM. Moreover, while it is indeed true that negative results in the MGGM are generally weaker than those in the SGGM and should be interpreted cautiously [Zha22, DHH+21], our result is a natural first step towards proving a stronger negative result for a basic question of whether 3-NIKE can be based merely on simple algebraic assumptions without pairing.

Discussion. In our third contribution, we prove our lower bound in Maurer's generic group model, whereas our positive result holds in *Shoup*'s generic group model, which is more flexible: this leaves a gap between our positive and negative results. We refer to [Zha22] for an in-depth discussion on the differences between these two models. We view as an interesting question the goal of closing the gap between our positive and negative results, either by building a 4-NIKE protocol with quadratic security in Maurer's generic group model, or by extending our impossibility result to Shoup's generic group model.

1.2 Technical Overview: Building NIKE in the ROM and GGM

We start by covering our positive results. Our starting point is the classical 2-party NIKE of Merkle in the random oracle model, which works as follows: let $H : [n^2] \mapsto \{0,1\}^\lambda$ (for security parameter λ) be an injective random oracle. Alice and Bob sample $(a_1, \cdots, a_n) \xleftarrow{\$} [n^2]^n$ and $(b_1, \cdots, b_n) \xleftarrow{\$} [n^2]^n$ respectively, and exchange the hashes of these values: Alice sends $(H(a_1), \cdots, H(a_n))$, and Bob sends $(H(b_1), \cdots, H(b_n))$. By the birthday paradox, with some constant probability, there will be a collision

[5] Our proof is for $K = 3$ which will directly imply the negative result for any $K \geq 3$.

$a_i = b_j$. Since H is injective, every hash collision corresponds to an input collision. Alice and Bob can identify (say) the first such collision, and set $\mathsf{key} \leftarrow a_i = b_j$ to be their shared key. To find the shared key, any adversary must essentially query the random oracle on $\Omega(n^2)$ positions, hence the protocol has fine-grained security with near-quadratic gap. More generally, any $n^{2-\varepsilon}$-query adversary has probability $n^{-\varepsilon}$ of querying the shared key; this probability can be reduced to negligible by letting Alice and Bob send $n \cdot \log n$ hashes instead, and identifying $\ell(n) = \omega(\log n)$ collisions, defining the key as the XOR of the ℓ keys.

Fine-grained multiparty NIKE in the ROM. In this work, we show that the above protocol can be directly generalized to the K-party setting, if we set the domain size of the random oracle to $n^{1+1/(K-1)}$: this guarantees that K random $n \cdot \log n$-sized tuples will have $\ell(n)$ K-collision with some constant probability. The security analysis essentially unchanged, and shows that $n^{1+1/(K-1)-\varepsilon}$-query adversaries have negligible probability of finding the final key. Correctness is slightly more technical, as it requires proving that the number of K-collisions among K random $n \cdot \log n$-sized tuples is at least $\ell(n)$ with overwhelming probability. It follows from a sequence of concentration bounds: we identify some s_1 such that with overwhelming probability, the number of collisions among the hashes of the first two parties is at least s_1 (s_1 can be computed by a straightforward Chernoff bound). Then, we identify s_2 such that with overwhelming probability, for any fixed set of s_1 values, there will be at least s_2 collisions between this set and the third party's hashes. We proceed this way, using a sequence of K Chernoff bounds to identify $s_1 > s_2 > \cdots > s_{K-1} = \ell(n)$ such that the number of K-collisions is at least s_{K-1} with overwhelming probability. From here, correctness follows immediately from the injectivity of the oracle. Even though security against $n^{1+1/(K-1)-\varepsilon}$-query adversaries gets worse as K grows, for every constant K, it still shows a polynomial gap between the honest parties' running time and that of the adversary.

Fine-grained 4-NIKE from Idealized 2-NIKE. The above protocol achieves K-NIKE, at the cost of strongly restricting the adversarial runtime: even for $K = 3$, the protocol only withstands $o(n^{1.5})$-time adversaries. However, since we only used a random oracle (*i.e.* an idealized hash function), one could reasonably hope that a better gap can be achieved if we start from stronger 'public key' primitives. As a starting point, we describe a construction of a 4-party NIKE starting from an idealized 2-NIKE oracle. While this construction does not directly yield a candidate classical instantiation (unlike ROM-based construction, which yield heuristic instantiation using a hash function), it captures the core intuition of our next construction, while abstracting out some of the technicalities. Concretely, we consider the following idealized 2-NIKE oracle with two procedures:

- $\mathsf{Msg} : [N] \mapsto \{0,1\}^*$ is an injective random oracle over the domain $[N]$.
- $\mathsf{Key} : [N] \times \{0,1\}^* \mapsto \{0,1\}^\lambda$, on input an element r of the domain $[N]$, and a bit-string s, it checks whether $s = \mathsf{Msg}(\mathrm{r}')$ for some r'. If there is such an r', it returns $h(\mathrm{r}_0, \mathrm{r}_1)$, where $(\mathrm{r}_0, \mathrm{r}_1)$ is a lexicographic ordering of $(\mathrm{r}, \mathrm{r}')$ and h is a random function from $[N] \times [N] \mapsto \{0,1\}^\lambda$.

Relative to $(\mathsf{Msg}, \mathsf{Key})$, it is straightforward to see that there exists an ideally-secure 2-NIKE scheme as follows: Alice and Bob broadcast $\mathsf{m_A} = \mathsf{Msg}(\mathrm{r_A})$ and $\mathsf{m_B} = \mathsf{Msg}(\mathrm{r_B})$ respectively, and obtain a shared key $\mathsf{key} = h(\mathrm{r_A}, \mathrm{r_B}) = \mathsf{Key}(\mathrm{r_A}, \mathsf{m_B}) = \mathsf{Key}(\mathrm{r_B}, \mathsf{m_A})$. Furthermore, interestingly there also exists a 4-NIKE relative to $(\mathsf{Msg}, \mathsf{Key})$ over domain $[N] = [n^2]$, with quadratic hardness gap (improving upon the collision-based approach of our construction of K-NIKE in the ROM) as follows. Fix four parties $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \mathsf{P}_4)$. At a high level, the protocol proceeds by (1) letting $(\mathsf{P}_1, \mathsf{P}_2)$ agree on a common randomness r_{12} with associated message $\mathsf{m}_{12} = \mathsf{Msg}(\mathrm{r}_{12})$ by looking for a randomness collision, (2) letting $(\mathsf{P}_3, \mathsf{P}_4)$ agree on $(\mathrm{r}_{34}, \mathsf{m}_{34})$ via the same collision-finding procedure, and (3) letting $(\mathsf{P}_1, \mathsf{P}_2)$ and $(\mathsf{P}_3, \mathsf{P}_4)$ play the roles of Alice and Bob respectively and derive a shared key using the Key oracle. More precisely:

1. Each party P_i samples n random elements $(\mathrm{r}_1^{(i)}, \cdots, \mathrm{r}_n^{(i)}) \xleftarrow{\$} [n^2]$ and broadcasts $(\mathsf{m}_1^{(i)}, \cdots, \mathsf{m}_n^{(i)}) = (\mathsf{Msg}(\mathrm{r}_1^{(i)}), \cdots, \mathsf{Msg}(\mathrm{r}_n^{(i)}))$.
2. With some constant probability, there exists two positions j_0, j_1 such that $\mathrm{r}_{j_0}^{(1)} = \mathrm{r}_{j_1}^{(2)}$, leading to a hash collision. P_1 and P_2 identify this collision; let r_{12} denote the collision, and m_{12} denote the corresponding message.
3. Similarly, P_3 and P_4 identify a collision r_{34} with message m_{34} among their vectors of messages.
4. P_1 and P_2 output $\mathsf{Key}(\mathrm{r}_{12}, \mathsf{m}_{34})$, and P_3 and P_4 output $\mathsf{Key}(\mathrm{r}_{34}, \mathsf{m}_{12})$.

Correctness follows easily by inspection. For security, any adversary that manages to find the common key key with non-negligible advantage must have queried Key on either $(\mathrm{r}_{34}, \mathsf{m}_{12})$ or $(\mathrm{r}_{12}, \mathsf{m}_{34})$. Without loss of generality, we can assume that the adversary always queries message with its first input to Key: therefore, the adversary must have queried either r_{12} or r_{34} to Msg. By the same analysis as for Merkle puzzles, an $O(n^{2-\varepsilon})$-query adversary can find such a query with probability at most $n^{-\varepsilon}$. As before, one can reduce the adversary's advantage to negligible by generating $\ell(n) = \omega(\log n)$ collisions per pair of party instead, and defining the shared keys to be the XOR of the $\ell(n)$ outputs of Key.

Fine-grained 4-NIKE in the SGGM. With the above template in mind, a natural idea is to replace the idealized oracle $(\mathsf{Msg}, \mathsf{Key})$ by a Diffie-Hellman key exchange, to get a 4-party NIKE over Diffie-Hellman groups with quadratic security gap. Unfortunately, this does not work! To see the issue, let us fix a cyclic group $\mathbb{G}$ of size n^2, with a generator g. Replacing $(\mathsf{Msg}, \mathsf{Key})$ by a Diffie-Hellman key exchange, we get the following (1st try) protocol:

1. Each party P_i samples n random elements $(\mathrm{r}_1^{(i)}, \cdots, \mathrm{r}_n^{(i)}) \xleftarrow{\$} [n^2]$ and broadcasts $(\mathsf{m}_1^{(i)}, \cdots, \mathsf{m}_n^{(i)}) = (g^{\mathrm{r}_1^{(i)}}, \cdots, g^{\mathrm{r}_n^{(i)}})$.
2. With some constant probability, there exists two positions j_0, j_1 such that $\mathrm{r}_{j_0}^{(1)} = \mathrm{r}_{j_1}^{(2)}$, leading to a collision between the group elements. P_0 and P_1 identify this collision; let r_{12} denote the collision, and m_{12} denote the corresponding message.
3. Similarly, P_3 and P_4 identify a collision r_{34} with message m_{34} among their vectors of messages.
4. P_1 and P_2 output $\mathsf{key} \leftarrow (\mathsf{m}_{34})^{\mathrm{r}_{12}}$, and P_3 and P_4 output $\mathsf{key} \leftarrow (\mathsf{m}_{12})^{\mathrm{r}_{34}}$.

The above protocol, however, turns out to be completely broken! The adversary can compute the discrete logarithm (in base g) of any group element in time $\sqrt{n^2} = n$, using a standard generic algorithm (e.g. Shank's baby-step giant-step algorithm [Sha71], or Pollard's rho algorithm [Pol75]). Hence, the adversary can recover $\mathrm{r}_{12} = \mathsf{dlog}_g(\mathsf{m}_{12})$ in time n and recompute the shared key.

Above, the issue is that our 4-NIKE from an idealized 2-NIKE crucially relied on its optimal security: it must be secure over a size-n^2 domain, when the honest parties can run in time n. However, over cryptographic groups, one can always get a quadratic speedup over naive brute-force. Fortunately, there is a way around. Our key idea is the following: we increase the group size to $|\mathbb{G}| = n^4$, so that generic discrete logarithm now takes $\Omega(n^2)$-time. Doing so, we strongly reduced the probability that the honest parties can find a collision among length-n vectors of group elements. To get around this issue, we make two important observations:

1. Although there will not be any full collision, we can guarantee that there will be a *prefix-collision* with some constant probability: a pair of group elements whose exponent share *the same first half*.
2. Assume that two parties identified a group element g^x such that (a) one of the parties knows x, and (b) the other party knows *the first half of the bits of* x. Then the second party can recover x entirely in time $O(n)$: there are only $\approx n^2$ possible exponents x consistent with the prefix known to the party. Furthermore, known generic discrete logarithm algorithms actually run in time *square root of the search space* when the exponent search space is an interval. Therefore, the party can recover x using, e.g., Pollard's rho algorithm in $O(n)$ time.

At a high level, our protocol combines (1) a Merkle-style 2-NIKE to identify a prefix-collision (hence using the ROM) and (2) a generic discrete logarithm computation running in time $\sqrt{T}$ for solving discrete logarithms with exponents in an interval of size T. Concretely, let $H : [n] \mapsto \{0,1\}^*$ be an injective random oracle, and let $\mathbb{G}$ be a cyclic group of prime order p with $p \approx n^4$ (we assume that the order is exactly n^4 below to simplify the description). Then, our actual protocol proceeds as follows.

1. P_1 samples n exponents $(\mathrm{r}_1, \cdots, \mathrm{r}_n) \xleftarrow{\$} [n^4]^n$. For each $i \le n$, write $\mathrm{r}_i = a_i + n^2 \cdot b_i$ with $(a_i, b_i) \in [n^2] \times [n^2]$. P_0 broadcasts $(s_1, \cdots, s_n) \leftarrow (g^{\mathrm{r}_1}, \cdots, g^{\mathrm{r}_n})$ and $(H(a_1), \cdots, H(a_n))$.
2. P_2 samples n values $(a'_1, \cdots, a'_n) \xleftarrow{\$} [n^2]^n$ and broadcasts $(H(a'_1), \cdots, H(a'_n))$. This lets P_1 and P_2 identify a collision $a_i = a'_j$ with a constant probability.
3. P_2 computes $g^{n^2 \cdot b_i} = s_i / g^{a'_j}$. Note that $b_i \in [n^2]$. Hence, P_2 recovers b_i (and therefore $\mathrm{r}_i = a'_j + b_i \cdot n^2$) in time $O(n)$ by computing the discrete logarithm of $g^{n^2 \cdot b_i}$ in base g^{n^2} using e.g.

Pollard's algorithm [Pol75][6]. At this stage, P_1 and P_2 agree on a common pair $(r, s) \leftarrow (\mathrm{r}_i, s_i)$ (and i, s_i are public).
4. P_3 and P_4 do similarly, and agree on (r', s') where s' is publicly known.
5. P_1 and P_2 output $(s')^r$, while P_3 and P_4 output $\mathsf{m}^{r'}$.

Correctness follows easily by inspection. In the body of the paper, we prove security in Shoup's generic group model (SGGM) together with the injective random oracle model. Using the random oracle, however, is merely for the ease of presentation, as the SGGM implies the existence of an injective random oracle, hence we get a 4-party NIKE with quadratic security gap in the SGGM.

Above, there is nothing specific to the 4-party setting: given a generic group $\mathbb{G}$ equipped with a $(K-1)$-linear map e, K parties can agree on a common key by each broadcasting a random group element $g_i = g^{\mathrm{r}_i}$, and outputting $\mathsf{key} = e((g_j)_{j \neq i})^{\mathrm{r}_i}$. Then, the above construction allows pairs of parties to agree on a common input (r_i, g_i) to this K-NIKE with a group of size $|\mathbb{G}| = n^4$ using $O(n)$ communication in a single round of interaction. Therefore, this yields a $2K$-NIKE with quadratic security gap in the generic $(K-1)$-linear group model. In particular, we can obtain a 6-NIKE with quadratic security in the generic bilinear group model.

1.3 Technical Overview: Breaking 3-NIKE in Maurer's GGM

We also prove a limitation of how much algebraic structure, without pairing, can help building K-NIKE protocols for $K > 2$. In particular, we prove that in *Maurer's* generic group model (MGGM), where the access to the group is further limited through an oracle who does all the calculations, one cannot achieve more than quadratic gap between the honest parties and the adversary's query complexity. Recall that in MGGM, each party P has an array Arr_P that does the following operations: (1) Arr_P stores group elements from $\mathbb{G}$, starting with 1 written at the beginning, (2) it adds them (Add operation) when P asks Arr_P to do so, and stores the result at the end of the array, and (3) it can provide zero-tests (Zero operation) for all the stored group elements. We actually need to work with a generalization of this model in which parties can *exchange* group elements *directly* through their oracles (as group elements do not have an explicit representation). See Definition 14 for details. Finally, when it comes to key-agreement in MGGM, without loss of generality, we ask the parties to agree on a group element written in their oracle (see Remark 16.)

Here we highlight the key ideas in our attack on any 3-NIKE protocol Π in the MGGM. Our proof can be best explained in two steps: (1) Breaking Π, assuming that the honest parties do not ask any Zero queries. (2) Breaking Π, even if parties ask Zero queries by reducing this task to the case of protocols without Zero queries. Below, we explain both of these steps and their corresponding ideas.

Structure of the key. For simplicity suppose $\mathsf{A}, \mathsf{B}, \mathsf{C}$, as part of Π, agree on a key key with probability 1. Let us focus on A and analyze the structure of the group element key_A that it produces as its key (see Definition 14). This key is a function of Alice's randomness r_A, the transcript tran and the group elements that Alice receives from Bob ($\overline{\mathrm{q}}_\mathsf{B} = (\mathrm{q}_{\mathsf{B},1}, \cdots, \mathrm{q}_{\mathsf{B},\gamma})$) and Charlie ($\overline{\mathrm{q}}_\mathsf{C} = (\mathrm{q}_{\mathsf{C},1}, \cdots, \mathrm{q}_{\mathsf{C},\gamma})$). Since Alice's algorithm is in the MGGM, the key key will, therefore, be a linear function of the components of $\overline{\mathrm{q}}_\mathsf{B}, \overline{\mathrm{q}}_\mathsf{C}$ with coefficients $\overline{\mathrm{a}}_\mathsf{B}, \overline{\mathrm{a}}_\mathsf{C}$ that can arbitrarily depend on Alice's randomness r_A and transcript tran (see Lemma 17). In particular, $\mathsf{key}_\mathsf{A} = \overline{\mathrm{a}}_\mathsf{B} \cdot \overline{\mathrm{q}}_\mathsf{B} + \overline{\mathrm{a}}_\mathsf{C} \cdot \overline{\mathrm{q}}_\mathsf{C}$, where $\cdot$ is inner product.

Breaking 3-NIKEs without zero tests: randomness switching lemma. Suppose E starts by *re-sampling* Charlie's randomness into r'_C conditioned on the message m_C. This change will lead to a different set of group elements $\overline{\mathrm{q}}'_\mathsf{C} = (\mathrm{q}'_{\mathsf{C},1}, \cdots, \mathrm{q}'_{\mathsf{C},\gamma})$ broadcast by C, and hence Alice's key will change as well into $\mathsf{key}'_\mathsf{A} = \overline{\mathrm{a}}_\mathsf{B} \cdot \overline{\mathrm{q}}_\mathsf{B} + \overline{\mathrm{a}}_\mathsf{C} \cdot \overline{\mathrm{q}}'_\mathsf{C}$. However, a crucial point is that the component $\overline{\mathrm{a}}_\mathsf{B} \cdot \overline{\mathrm{q}}_\mathsf{B}$ in this linear function *stays the same*. This important point is directly enabled by the fact that there are *three* parties involved, and we would not have this property in the 2-NIKE setting. Now, our attack will directly take advantage of this common part $\overline{\mathrm{a}}_\mathsf{B} \cdot \overline{\mathrm{q}}_\mathsf{B}$ in Alice's key when Charlie switches its randomness to r'_C. In particular, suppose $\mathsf{Key}_\mathsf{A}(\mathrm{r}'_\mathsf{A}, \mathrm{r}'_\mathsf{C})$ be Alice's key when Alice and Charlie use random seeds $\mathrm{r}'_\mathsf{A}, \mathrm{r}'_\mathsf{C}$ (that are compatible with the text messages sent by Alice and Bob), and Bob's randomness is fixed to its true randomness r_B. Further using the same observation above about switching the

[6] The seminal paper of Pollard describes two algorithms for solving discrete logarithms. The second, lesser known algorithm, usually called *Pollard's kangaroo algorithm*, solves discrete logarithms with exponents over intervals $[u, v]$ in time $\sqrt{v-u}$.

randomness of a party, we show that when *both* Alice and Charlie resample their random seeds $\mathsf{r}'_\mathsf{A}, \mathsf{r}'_\mathsf{C}$, conditioned on the shared text messages (ignoring the group elements that are encoded), the common parts in Alice's key across these new "executions" of the protocol will cancel out each other and we obtain the following invariant,

$$\mathsf{Key}_\mathsf{A}(\mathsf{r}_\mathsf{A}, \mathsf{r}_\mathsf{C}) = \mathsf{Key}_\mathsf{A}(\mathsf{r}_\mathsf{A}, \mathsf{r}'_\mathsf{C}) + \mathsf{Key}_\mathsf{A}(\mathsf{r}'_\mathsf{A}, \mathsf{r}_\mathsf{C}) - \mathsf{Key}_\mathsf{A}(\mathsf{r}'_\mathsf{A}, \mathsf{r}'_\mathsf{C}),$$

which is formalized in Lemma 19 as the randomness switching lemma. Finally, the eavesdropper attacker can directly use the above equality to use re-sampled keys $\mathsf{r}'_\mathsf{A}, \mathsf{r}'_\mathsf{B}$ (which she obtains using inverse sampling) to obtain all three fake keys on the right hand side, using which it can obtain the true key.

Breaking 3-NIKEs *with* zero tests: learning useful linear tests. We now describe the extra ideas needed to handle zero tests done by the parties. Firstly, we can assume without loss of generality that all zero tests by $\mathsf{A}, \mathsf{B}, \mathsf{C}$ are asked *after* they receive the text messages and the group elements (through their oracles). In particular, parties' randomness will directly determine their messages and the group elements they send. Eve, has direct access only to the text messages sent by the parties, because the group elements are encoded. However, Eve can also perform zero-test queries over the (vector of) group elements $\overline{\mathsf{q}} = (\overline{\mathsf{q}}_\mathsf{A}, \overline{\mathsf{q}}_\mathsf{B}, \overline{\mathsf{q}}_\mathsf{C})$. Eve's goal here is to learn any *useful* zero-test query over $\overline{\mathsf{q}}$ such that the answer to the zero-test queries by $\mathsf{A}, \mathsf{B}, \mathsf{C}$ will follow. In particular, we say that a set of *linear constraints* LinCon (containing both linear equalities and inequalities) over $\overline{\mathsf{q}}$ are $(1-\varepsilon)$-*useful* if the following two properties hold.

1. *Pure restrictions.* We say that LinCon is pure if all of its constraints are over an *individual* party. Namely, for each constraint c in LinCon, there is a party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ such that c is a constraint over $\overline{\mathsf{q}}_\mathsf{P}$.
2. *Covering heavy zero tests.* We say that $(\mathsf{LinCon}, \mathsf{tran})$ covers ε-*heavy* zero-tests, if for every zero test query f (over the variables $\overline{\mathsf{q}} = (\overline{\mathsf{q}}_\mathsf{A}, \overline{\mathsf{q}}_\mathsf{B}, \overline{\mathsf{q}}_\mathsf{C})$) whose answer is not implied by the equalities in LinCon (f is not trivially positive or negative using the equalities in LinCon), the probability of answering positively is at most ε over the randomness of the parties.

We first explain how we find useful sets of linear constraints. We then explain why finding them can be used for a successful attack by reduction to the setting of protocols with no zero tests.

Finding a useful set. Finding a useful set is rather straight forward. Eve will iteratively pick any *pure* zero test query (i.e., only dealing with one party's shared group elements) over $\overline{\mathsf{q}}$ that is both ε-heavy to hold (positively) and that it is *not* in the span of the equalities already in LinCon. Since the dimension of the linear constraints over LinCon is limited by 3γ (i.e., the total number of group elements shared by the parties) this process will stop in about $3\gamma/\varepsilon$ steps. The proof is similar to the proof of efficiency of the heavy-learner of [BMG07, BM17].

Using a useful set. If a set LinCon is useful, then by the first (pure restrictions) property, it imposes a *product* distribution over the randomness of $\mathsf{A}, \mathsf{B}, \mathsf{C}$. Therefore, one can define an imaginary protocol with respect to the fixed text messages tran and LinCon, in which $\mathsf{A}, \mathsf{B}, \mathsf{C}$ will pick their random seeds conditioned on $(\mathsf{tran}, \mathsf{LinCon})$ and run their key extraction algorithms to agree on a key. Furthermore, by the second (light tests) property, assuming ε is sufficiently small, with high probability *all* of the zero-test queries of $\mathsf{A}, \mathsf{B}, \mathsf{C}$ will be answered merely by LinCon, or that they will be answered negatively. Therefore, all such queries could be *removed* from the protocol, and we will be back to a protocol *without* any zero tests. This means that we are back to the simpler case of no zero tests, which was resolved already.

To see why the zero test queries can be compiled out of the protocol (conditioned on $(\mathsf{tran}, \mathsf{LinCon})$), in the following for simplicity suppose Alice asks only one zero-test query. By the first (pure restrictions) property, one can fix this zero-test query without any further restriction on the distribution of the random seeds of Bob or Charlie. Therefore, all we need to show is that the answer to this (non-trivial) zero-test query will be negative with probability $1-\varepsilon$. This would be the case if we had learned all the ε-heavy linear constraints that deal with the variables in *both* of $\overline{\mathsf{q}}_\mathsf{B}, \overline{\mathsf{q}}_\mathsf{C}$, while our set LinCon only contains heavy constraints that are pure. However, interestingly, one more application of the purity property shows that what we have learned in LinCon is already enough. In particular, we prove that, because of the independence of the distributions of $\mathsf{r}_\mathsf{B}, \mathsf{r}_\mathsf{C}$ conditioned on $(\mathsf{tran}, \mathsf{LinCon})$, the existence of any unlearned ε-heavy zero-test (not spanned by the equalities in LinCon) over $\overline{\mathsf{q}}_\mathsf{B}, \overline{\mathsf{q}}_\mathsf{C}$ will automatically imply the existence of an unlearned pure ε-heavy linear restriction on *either* of $\overline{\mathsf{q}}_\mathsf{B}, \overline{\mathsf{q}}_\mathsf{C}$. But such heavy restrictions are already learned by Eve!

2 Preliminaries

Definition 1 (K-Party Non-Interactive Key-Exchange (K-NIKE)). *For a security parameter λ, and a set of K parties $\{\mathsf{P}_1, \cdots, \mathsf{P}_K\}$, a K-party non-interactive key-exchange protocol for key space $\mathcal{KS}$ consists of a pair of algorithms $(\mathsf{Msg}, \mathsf{Key})$ defined as follows:*

- $\mathsf{Msg}(1^\lambda, i, \mathrm{r}_i) \to \mathsf{m}_i$*: The message generation algorithm takes as input the security parameter λ, an index $i \in [K]$ indicating the party P_i, and a randomness $\mathrm{r}_i \in \{0,1\}^\lambda$, and outputs the corresponding message $\mathsf{m}_i \in \{0,1\}^\lambda$. It is assumed that in the time of generating the messages, each party generates its message with the algorithm Msg and broadcasts it.*
- $\mathsf{Key}(i, \mathrm{r}_i, \mathsf{tran}) \to \mathsf{key}_i$*: The key generation algorithm takes as input the index $i \in [K]$ indicating the party P_i, P_i's randomness r_i and the transcript $\mathsf{tran} := (\mathsf{m}_j)_{j\in[K]}$ consisting of all the broadcasted messages from the time of message generation, and outputs a (shared) key $\mathsf{key}_i \in \{0,1\}^m$.*

A K-NIKE protocol satisfies the following properties:

- ***Correctness:*** *We say the scheme has completeness error $\delta(\lambda)$, if for all security parameters $\lambda \in \mathbb{N}$, all indices $i, j \in [K]$, and all choices of randomnesses r_i and r_j,*

$$\Pr_{\mathrm{r}_1,\ldots,\mathrm{r}_K}[\mathsf{Key}(i, \mathrm{r}_i, \mathsf{tran}) = \mathsf{Key}(j, \mathrm{r}_j, \mathsf{tran})] \geq 1 - \delta(\lambda),$$

  where $\mathsf{tran} = (\mathsf{m}_1, \ldots, \mathsf{m}_K)$ for $\mathsf{m}_i \leftarrow \mathsf{Msg}(1^\lambda, i, \mathrm{r}_i)$. We simply say the scheme is complete, if it has completeness error $\delta(\lambda) \leq \mathsf{negl}(\lambda)$. We say the scheme has perfect completeness if $\delta(\lambda) = 0$.
- ***Security:*** *For all security parameters $\lambda \in \mathbb{N}$, and all efficient adversaries $\mathcal{A}$, we define the advantage of $\mathcal{A}$ as follows:*

$$\mathrm{Adv}_{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{key}_1 = \mathsf{key}_{\mathcal{A}}] - \frac{1}{|\mathcal{KS}|}\right|$$

  where $\mathsf{key}_{\mathcal{A}} \leftarrow \mathcal{A}(1^\lambda, \mathsf{tran})$ and $\mathsf{key}_1 \leftarrow \mathsf{Key}(1, r_1, \mathsf{tran})$, in which $\mathsf{tran} := (\mathsf{m}_i)_{i\in[K]}$ for $\mathsf{m}_i \leftarrow \mathsf{Msg}(1^\lambda, i, \mathrm{r}_i)$ for all $i \in [K]$. A K-NIKE protocol is secure if $\mathrm{Adv}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(n)$.

Note that randomnesses, messages, and keys can be viewed as vectors in the above definition. We can also define a fine-grained variant of K-NIKE as follows.

Definition 2 ((t,ε)-Secure K-NIKE). *For functions $t = t(\cdot), \varepsilon = \varepsilon(\cdot)$ a (fine-grained) (t,ε)-secure K-NIKE has the same syntax as the subroutines $\mathsf{Msg}, \mathsf{Key}$ in Definition 1, with the following additional conditions on its correctness and security properties. There is a function $n = n(\lambda)$, such that:*

- ***Correctness:*** *All parties (i.e., both $\mathsf{Msg}, \mathsf{Key}$ algorithms) run in time $\tilde{O}(n)$.*
- ***Security:*** *We only limit ourselves to adversaries who run in time $t(n)$, and for all such adversary their advantage shall be at most $\varepsilon(n)$.*

In other words, after changing the security parameter to n, honest parties run in quasi-linear time, while the adversary needs time t to gain advantage ε. When the protocol is in an idealized model, we use the number of queries by the algorithms to the oracle as the measure of their running time.

Definition 3 (Shoup's Generic Group Model (SGGM)). *Let $p \in \mathbb{Z}$ be a positive integer. For such fixed p, in Shoup's Generic Group Model (SGGM) all parties have access to an oracle with the following queries.*

- enc **query.** *Suppose S is a* label space *of size $|S| \geq p$, and let enc be a random injective function from $\mathbb{Z}_p$ to S.*
- Add **query.** *If $z = c_1 \cdot x + c_2 \cdot y$, then $\mathsf{Add}(c_1, c_2, \mathsf{enc}(x), \mathsf{enc}(y)) = \mathsf{enc}(z)$.*

In this paper, for simplicity of presentation, we denote $\mathsf{enc}(x)$ by g^x, even when we are in the generic group model and g^x is not an actual exponentiation.

Random Oracle Model. Recall that in the Random Oracle Model ROM, all parties have access to a function H randomly sampled from the set $\mathcal{H}$ of all functions $f : \mathcal{N} \to \mathcal{M}$, and the input/output spaces $\mathcal{N}, \mathcal{M}$ are chosen differently in different contexts. These variants can simulate each other, but when it comes to fine-grained efficiency and security properties, the choice of random oracle can be more important. In this model, we primarily count the number of oracle queries as the substitute for "running time."

Lemma 4 (Chernoff Bound). *Let $X = \Sigma_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}[X] = \Sigma_{i=1}^{n} p_i$. Then*

1. ***Upper Tail:*** $\Pr[\mathbf{X} \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2}{2+\delta}\mu}$ *for all* $\delta > 0$;
2. ***Lower Tail:*** $\Pr[\mathbf{X} \leq (1-\delta)\mu] \leq e^{-\mu\frac{\delta^2}{2}}$ *for all* $0 < \delta < 1$.

3 3-NIKE in the Random Oracle Model

In this section, we construct a 3-party non-interactive key-exchange protocol in the random oracle model with non-trivial fine-grain security by generalizing Merkle Puzzles [Mer74]. To give a high-level idea of how the protocol works, we start with a similar idea of the classical 2-party NIKE of Merkle in random oracle model. Namely, for a security parameter λ and a given random oracle H, each of the three parties samples a set of secret values $\bar{\mathsf{r}}_i$ of size $\lambda^2\ell(\lambda)$, computes $H(\bar{\mathsf{r}}_i)$, which is the output of the random oracle on each value of the set, and broadcasts it. It can be shown that with high probability, there will be a set of collisions of size at least $\ell(\lambda)$. Then, *without any further interaction*, the parties pick the first lexicographic $\ell(\lambda)$ collisions, or any other natural way of pre-agreeing on which subset to pick, and compute the shared key similar to the 2-party NIKE protocol of Merkle.

Construction 5 (3-NIKE ROM-Based Protocol) *For a security parameter λ, let $H : [\lambda^3] \to \{0,1\}^\lambda$ be a random oracle, and $\ell(\lambda) = \log^2(\lambda)$ a minimal intersection size parameter. The 3-NIKE protocol between parties $\{\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3\}$ would be as follows:*

- $\mathsf{Msg}(1^\lambda, i, \bar{\mathsf{r}}_i) \to \overline{\mathsf{m}}_i$: *For each party P_i, on input the security parameter λ and P_i's randomnesses $\bar{\mathsf{r}}_i$, which is viewed as a set $\bar{\mathsf{r}}_i \subset [\lambda^3]$ of size $\lambda^2 \cdot \log(\lambda)$, the message generation algorithm proceeds as follows:*
  1. *View $\bar{\mathsf{r}}_i$ as $\{\mathsf{r}_{ij}\}_{j \in \lambda^2 \cdot \log(\lambda)}$, and compute $H(\bar{\mathsf{r}}_i) \coloneqq \{H(\mathsf{r}_{ij})\}_{j \in \lambda^2 \cdot \log(\lambda)}$.*
  2. *Output and broadcast the set of messages as $\overline{\mathsf{m}}_i \coloneqq H(\bar{\mathsf{r}}_i)$.*
- $\mathsf{Key}(i, \bar{\mathsf{r}}_i, \mathsf{tran}) \to \overline{\mathsf{key}}_i$: *On input an index $i \in [3]$, the party P_i's randomnesses $\bar{\mathsf{r}}_i$, and the transcript $\mathsf{tran} \coloneqq (\overline{\mathsf{m}}_j)_{j \in [3]/\{i\}}$, the key generation algorithm proceeds as follow:*
  1. *Invoke the message generation algorithm to obtain $\overline{\mathsf{m}}_i \leftarrow \mathsf{Msg}(1^\lambda, i, \bar{\mathsf{r}}_i)$. If $|\cap_{j \in [3]} \overline{\mathsf{m}}_j| < \ell(\lambda)$, the algorithm outputs 0 and aborts.*
  2. *Let $c_1, \cdots, c_{\ell(\lambda)}$ be the first $\ell(\lambda)$ lexicographic common outputs of $\cap_{j \in [3]} \overline{\mathsf{m}}_j$.*
  3. *All parties $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3$ are able to find the common inputs $s_1, \cdots, s_{\ell(\lambda)} \in \cap_{j \in [3]} \bar{\mathsf{r}}_j$ such that $H(s_i) = c_i$.*
  4. *The shared (output) key will be $\overline{\mathsf{key}}_i \coloneqq \bigoplus_{i=1}^{\ell(\lambda)} s_i$.*

In this section, we prove the following theorem.

Theorem 6. *Construction 5 is a (t, ε) 3-NIKE, where $t = n^{1.5}, \varepsilon = \mathsf{negl}(n)$.*

Proof. Let $n = \lambda^2$. The proofs of correctness and security are as follows.

Correctness. Due to $n = \lambda^2$, it holds that the algorithms running in $\tilde{O}(n)$.

We have three sets $\overline{\mathsf{m}}_1$, $\overline{\mathsf{m}}_2$, and $\overline{\mathsf{m}}_3$ of size $\lambda^2 \log(\lambda)$ chosen randomly from the set $[\lambda^3]$. It is easy to see that since we are in the random oracle model, if the Key algorithm does not abort, all three parties will receive the same key with probability 1. Therefore, for $\ell(\lambda) = \log^2(\lambda)$, in order to show that the parties will successfully obtain one shared key with high probability after running the algorithms properly, it suffices to show that the probability of the Key algorithm aborts is negligible. More formally, we want to show

$$\Pr\left[|\cap_{j \in [3]} \overline{\mathsf{m}}_j| \leq \ell(\lambda)\right] = \Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| \leq \ell(\lambda)] \leq \mathsf{negl}(\lambda).$$

In order to prove this, we adopt the ideas used in [BGI08] in a similar context/goal (about amplifying security in a two-party key agreement) and use a chain of Chernoff bounds. We know

$$\Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| \leq \ell(\lambda)] \leq 1 - \Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda)] \tag{1}$$

As a first step, it is easy to see that for every choice of $s \in [\lambda^2 \log(\lambda)]$, we have

$$\Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda)] \geq \Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \wedge |\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s]$$

$$= \Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s] \cdot \Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s]. \tag{2}$$

Now we can analyze $\Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s]$ for some $s \in [\lambda^2 \log(\lambda)]$, using a Chernoff bound similar to [BGI08]. Then viewing $\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2$ as a set, we can use the same idea for analyzing $\Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s]$. Let

$$\overline{\mathsf{m}}_1 = \{a_1, \cdots, a_{\lambda^2 \log(\lambda)}\}, \overline{\mathsf{m}}_2 = \{b_1, \cdots, b_{\lambda^2 \log(\lambda)}\}, \overline{\mathsf{m}}_3 = \{c_1, \cdots, c_{\lambda^2 \log(\lambda)}\}.$$

Now for analyzing the second probability in Equation 2 (i.e. $\Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s]$,) let X_i be the event that $b_i \in \overline{\mathsf{m}}_1$ (i.e. $b_i \in \overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2$.) Therefore,

$$\mu \coloneqq \mathbb{E}\left[\sum_{i=1}^{\lambda^2 \log(\lambda)} \mathbf{X}_i\right] = \frac{\lambda^2 \log(\lambda)}{\lambda^3} \cdot \lambda^2 \log(\lambda) = \lambda \cdot \log^2(\lambda).$$

Based on the way the event X_i is defined, we can view $|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2|$ as $\sum_{i=1}^{\lambda^2 \log(\lambda)} \mathbf{X}_i$,

$$\Pr\left[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s\right] = 1 - \Pr[\sum_{i=1}^{\lambda^2 \log(\lambda)} \mathbf{X}_i < s].$$

By lower tail of Chernoff bound in Lemma 4,

$$\Pr\left[\sum_{i=1}^{\lambda^2 \log(\lambda)} \mathbf{X}_i < (1-\delta)\lambda \log^2(\lambda)\right] \leq e^{-\frac{\delta^2}{2}\lambda \log^2(\lambda)}.$$

This concludes that

$$\Pr\left[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s\right] \geq 1 - e^{-\frac{\delta^2}{2}\lambda \log^2(\lambda)}. \tag{3}$$

For analyzing the first probability in Equation 2, we first simplify the probability as follows. Let $\overline{\mathsf{m}} \coloneqq \overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2$,

$$\begin{aligned} \Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s] &= \Pr[|\overline{\mathsf{m}} \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}| \geq s] \\ &\geq \Pr[|\overline{\mathsf{m}} \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}| = s]. \end{aligned} \tag{4}$$

Now, using a similar approach as before, we can find a lower bound for the complement of the above probability which will give us an upper bound for the first part of Equation 2. Namely,

$$1 - \Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| = s] = 1 - \Pr[|\overline{\mathsf{m}} \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}| = s]$$

Now, letting $\overline{\mathsf{m}} = \{m_1, \ldots, m_s\}$ and Y_i be the event that $c_i \in \overline{\mathsf{m}}$ (i.e. $c_i \in \overline{\mathsf{m}} \cap \overline{\mathsf{m}}_3$), we will have

$$\mu' \coloneqq \mathbb{E}\left[\sum_{i=1}^{\lambda^2 \log(\lambda)} \mathbf{Y}_i\right] = \frac{s}{\lambda^3}(\lambda^2 \log(\lambda)).$$

Using another Chernoff bound from Lemma 4,

$$\Pr\left[\sum_{i=1}^{\lambda^2 \log(\lambda)} \mathbf{Y}_i \leq (1-\delta')\frac{s}{\lambda^3}(\lambda^2 \log(\lambda))\right] \leq e^{-\frac{\delta'^2}{2}\frac{s \log(\lambda)}{\lambda}},$$

which results in

$$1 - \Pr[|\overline{\mathsf{m}} \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}| = s] \leq e^{-\frac{\delta'^2}{2}\frac{s \log(\lambda)}{\lambda}}. \tag{5}$$

Using the above in Equation 4 give us

$$\Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda) \mid |\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2| \geq s] \geq 1 - e^{-\frac{\delta'^2}{2}\frac{s \log(\lambda)}{\lambda}}. \tag{6}$$

Set $\delta = \delta' = \frac{1}{2}$ and $s = \frac{\lambda}{2}\log^2(\lambda)$. Using equations 3 and 6 in Equation 2, we get

$$\begin{aligned}\Pr[|\overline{\mathsf{m}}_1 \cap \overline{\mathsf{m}}_2 \cap \overline{\mathsf{m}}_3| > \ell(\lambda)] &\geq (1 - e^{-\frac{\lambda}{8}\log^2(\lambda)})(1 - e^{-\frac{\log^3(\lambda)}{16}}) \\ &\geq 1 - 2e^{-\frac{\log^3(\lambda)}{16}} = 1 - \mathsf{negl}(\lambda).\end{aligned}$$

which concludes the proof.

Security. We need to show that adversaries who ask $o(\lambda^3)$ queries have $\mathsf{negl}(\lambda)$ advantage of finding the shared key based on the transcript. In particular, we want to show that, any such adversary $\mathcal{A}$ will likely not query at least one of intersection points used by the key generation algorithm to compute the key.

Let $k = \lambda^3/3$ be the number of queries that the adversary $\mathcal{A}$ makes to the oracle, and assume w.l.o.g. that $\mathcal{A}$ does not repeat queries. Denote adversary's i'th query by q_i and let X_{ij} be the event that $H(q_i) = c_j$ for some $j \in [\ell(\lambda)]$. For any $i \leq \lambda^3/3$ and any $q_i \notin \{q_1, \cdots, q_{i-1}\}$, $H(q_i)$ is distributed uniformly within the set $\{0,1\}^\lambda \setminus \{H(q_1), \cdots H(q_{i-1})\}$ from the view of $\mathcal{A}$ prior to q_i. Hence, for all i, j, $\Pr[X_{ij} = 1] \leq \frac{1}{2^\lambda - k}$, and letting $k' = 2^\lambda - k$ and $X_i = \sum_{j=1}^{\ell(\lambda)} X_{ij}$, for all $i \leq k$, $\Pr[\mathbf{X}_i = 1] \leq \frac{\ell(\lambda)}{k'}$ regardless of other X_i's. Let $p = \frac{\ell(\lambda)}{k'}$, and assume X_i's are independent and $\Pr[\mathbf{X}_i = 1] = p$. By using the upper tail of a Chernoff bound from Lemma 4 and letting $\mu \coloneqq \mathbb{E}\left[\sum_{i=1}^k \mathbf{X}_i\right] = kp$, we have,

$$\Pr\left[\sum_{i=1}^k \mathbf{X}_i \geq \ell(\lambda)\right] = \Pr\left[\sum_{i=1}^k \mathbf{X}_i \geq \frac{k'}{k} \cdot (k \cdot \frac{\ell(\lambda)}{k'})\right] = \Pr\left[\sum_{i=1}^k \mathbf{X}_i \geq \frac{k'}{k} \cdot \mu\right]$$

where for large enough λ, the probability that $\mathcal{A}$ queries the oracle for all the common inputs (i.e. finding the shared key) will be bounded by:

$$\Pr\left[\sum_{i=1}^k \mathbf{X}_i \geq \ell(\lambda)\right] \leq e^{-\Omega(\ell(\lambda))} = \mathsf{negl}(\lambda)$$

where the last equality comes from $\ell(\lambda) = \log^2(\lambda)$. In the event that $\mathcal{A}$ did not query some s_i to the oracle, $\mathcal{A}$'s output is wrong with probability $\frac{1}{2}$ since from its point of view, the value of $\bigoplus_{i=1}^{\ell(\lambda)} s_i$ is 0 or 1 with equal probability. In other words, letting Y be the event that $\sum_{i=1}^k \mathbf{X}_i \geq \ell(\lambda)$,

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A}}(\lambda) &\coloneqq \left|\Pr\left[\overline{\mathsf{key}}_1 = \mathsf{key}_{\mathcal{A}}\right] - \frac{1}{2}\right| \\ &= \left|\Pr\left[\overline{\mathsf{key}}_1 = \mathsf{key}_{\mathcal{A}} \mid \mathbf{Y}\right] + \Pr\left[\overline{\mathsf{key}}_1 = \mathsf{key}_{\mathcal{A}} \mid \neg\mathbf{Y}\right] - \frac{1}{2}\right| \\ &\leq e^{-\Omega(\ell(\lambda))} + \frac{1}{2} - \frac{1}{2} = \mathsf{negl}(\lambda).\end{aligned}$$

Therefore, for any $d < 3$, an $O(\lambda^d)$-bounded adversary cannot make more than $\frac{\lambda^3}{3}$ queries for large enough λ.
All in all, for any $d < 3$, any $O(\lambda^d)$-bounded adversary $\mathcal{A}$, when the probability is taken over the possibilities for the random function, $\mathcal{A}$ can guess the key only with negligible advantage. □

Remark 7. Construction 5 can be generalized into a k-NIKE with $\Omega(\lambda^{k/(k-1)})$ security. The protocol will be similar to the above 3-NIKE protocol with a few changes. It will use a random oracle $H : [\lambda^k] \to \{0,1\}^\lambda$, and the randomness of each party should be from the set $[\lambda^k]$ and of size $\tilde{o}(\lambda^{k-1})$. The rest of the construction and the proofs will be similar with minor changes (e.g. there is a need for more applications of Chernoff bounds.)

Remark 8. We believe that with a similar approach to the one in [BGI08], one should be able to extend this result to getting a 3-NIKE similar to our Construction 5 from an "almost 1-1 OWF" instead of using a random oracle. We leave such studies for future work.

4 4-NIKE in Shoup's Generic Group Model

In this section, we construct a 4-NIKE protocol with quadratic security in Shoup's generic group model. As explained in the introduction, our construction can be interpreted as first constructing a 4-NIKE using an ideal 2-NIKE oracle, and then "substituting" the 2-NIKE ideal oracle with Shoup's GGM.

In this section, we introduce a candidate 4-NIKE protocol in the Shoup's Generic Group Model. To give a high-level intuition of the protocol, the idea of finding a shared key between four parties $\{\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \mathsf{P}_4\}$ is as follows. All parties, similar to the previous section, will choose a secret set of random values and broadcast some message as its corresponding public values (the broadcast messages might differ for each party), such that after the interaction, P_1 and P_2 will be able to identify a single secret value where its corresponding public message has already been broadcast, and the same will hold for P_3 and P_4. From there, and *without any further interaction*, $(\mathsf{P}_1, \mathsf{P}_2)$ will act as a single party, and so as $(\mathsf{P}_3, \mathsf{P}_4)$. Therefore, one can simply run a 2-party NIKE between these two parties and without the need of an interaction.

Before going into details of the protocol, we should note that we use the fact that we can view $x \in [\lambda^4]$ as $(x_1, x_2) \in [\lambda^2] \times [\lambda^2]$ for two isomorphic groups of size $\theta(\lambda^4)$ and $\theta(\lambda^2) \times \theta(\lambda^2)$ throughout our construction and proofs.

Construction 9 *For a security parameter λ, let $H : [\lambda^2] \to \{0,1\}^\lambda$ be a random oracle, $\mathbb{G}$ be a generic group (in the sense of Shoup) of size $\theta(\lambda^4)$ whose encodings are denoted by the "generator" g. Let $\ell(\lambda) = \log^2(\lambda)$ be the minimal intersection size parameter. The NIKE protocol between parties $\{\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \mathsf{P}_4\}$ is as follows:*

- $\mathsf{Msg}(1^\lambda, i, \bar{\mathrm{r}}_i) \to \overline{\mathsf{m}}_i$*: For each party P_i, on input the security parameter λ and P_i's randomnesses $\bar{\mathrm{r}}_i$, which is a random set $\bar{\mathrm{r}}_i \subset [\lambda^4]$ of size $\lambda \cdot \ell(\lambda)$, the message generation algorithm proceeds as follows:*
  1. *View $\bar{\mathrm{r}}_i$ as $\{\mathrm{r}_{ij}\}_{j \in [\lambda \cdot \ell(\lambda)]}$, and for $j \in [\lambda \cdot \ell(\lambda)]$, view each element as $\mathrm{r}_{ij} := (\mathrm{r}_{ij}^{(1)}, \mathrm{r}_{ij}^{(2)}) \subset [\lambda^2] \times [\lambda^2]$, where $\mathrm{r}_{ij}^{(1)}$ and $\mathrm{r}_{ij}^{(2)}$ are the first and second half of the value r_{ij}, respectively.*
  2. *Compute $H(\bar{\mathrm{r}}_i^{(1)}) := \left(H(\mathrm{r}_{ij}^{(1)})\right)_{j \in [\lambda \cdot \ell(\lambda)]}$, and $g^{\bar{\mathrm{r}}_i} := (g^{\mathrm{r}_{ij}})_{j \in [\lambda \cdot \ell(\lambda)]}$.*
  3. *Output and broadcast the messages as $\overline{\mathsf{m}}_i = \left(H(\bar{\mathrm{r}}_i^{(1)}), g^{\bar{\mathrm{r}}_i}\right)$ for $i \in \{1,3\}$, and $\overline{\mathsf{m}}_i = H(\bar{\mathrm{r}}_i^{(1)})$ for $i \in \{2,4\}$.*
- $\mathsf{Key}(i, \bar{\mathrm{r}}_i, \overline{\mathsf{tran}}) \to \mathsf{key}_i$*: On input an index $i \in [4]$, the party P_i's randomnesses $\bar{\mathrm{r}}_i$, and the transcript $\overline{\mathsf{tran}} := (\overline{\mathsf{m}}_j)_{j \in [4]/\{i\}}$, the key generation algorithm proceeds as follows:*
  1. *Invoke the message generation and obtain $\overline{\mathsf{m}}_i \leftarrow \mathsf{Msg}(1^\lambda, i, \bar{\mathrm{r}}_i)$, and parse $\overline{\mathsf{m}}_l = \left(H(\bar{\mathrm{r}}_l^{(1)}), g^{\bar{\mathrm{r}}_l}\right)$ or $\overline{\mathsf{m}}_l = H(\bar{\mathrm{r}}_l^{(1)})$ based on whether $l = 1, 3$ or $l = 2, 4$.*
  2. *If either $|H(\bar{\mathrm{r}}_1^{(1)}) \cap H(\bar{\mathrm{r}}_2^{(1)})| = 0$ or $|H(\bar{\mathrm{r}}_3^{(1)}) \cap H(\bar{\mathrm{r}}_4^{(1)})| = 0$, the algorithm outputs 0 and aborts.*
  3. *Let c be the first lexicographic common output of the $H(\bar{\mathrm{r}}_1^{(1)}) \cap H(\bar{\mathrm{r}}_2^{(1)})$, and c' be the first lexicographic common output of $H(\bar{\mathrm{r}}_3^{(1)}) \cap H(\bar{\mathrm{r}}_4^{(1)})$.*
  4. *Parties P_1 and P_2 are able to find the common input $s \in \bar{\mathrm{r}}_1^{(1)} \cap \bar{\mathrm{r}}_2^{(1)}$ such that $H(s) = c$. And similarly, P_3 and P_4 are able to find the common input $s' \in \bar{\mathrm{r}}_3^{(1)} \cap \bar{\mathrm{r}}_4^{(1)}$ such that $H(s') = c'$ (there is no need of further interaction between the parties.)*
  5. *Note that s is the first half of the corresponding element in $\bar{\mathrm{r}}_1 \cap \bar{\mathrm{r}}_2$. Having $g^{\bar{\mathrm{r}}_1}$ from party P_1's message, party P_2 is able to find the other half of the corresponding element in the set $\bar{\mathrm{r}}_1$ from s with a baby-step giant-step (BSGS) or Pollard's rho algorithm. Similarly, having $g^{\bar{\mathrm{r}}_3}$ from party P_3's message, party P_4 is able to find the second half of the corresponding element in the set $\bar{\mathrm{r}}_3$ from s'.*
  6. *Now, P_1 and P_2 have a common randomness in the set $\bar{\mathrm{r}}_1$, and P_3 and P_4 have a common randomness in $\bar{\mathrm{r}}_3$. Let the common randomness of P_1 and P_2 be $\hat{s} \subset \bar{\mathrm{r}}_1$, and the common randomness of P_3 and P_4 be $\hat{s}' \subset \bar{\mathrm{r}}_3$.*
  7. *View P_1 and P_2 as one party $\mathsf{P}_{12} = (\mathsf{P}_1, \mathsf{P}_2)$ with the randomness $\mathrm{r}_{12} := \hat{s}$ and their associated message $\mathsf{m}_{12} := g^{\mathrm{r}_{12}} = g^{\hat{s}}$. Similarly, view P_3 and P_4 as one party $\mathsf{P}_{34} = (\mathsf{P}_3, \mathsf{P}_4)$ with the randomness $\mathrm{r}_{34} := \hat{s}'$ and their associated message $\mathsf{m}_{34} := g^{\mathrm{r}_{34}} = g^{\hat{s}'}$. P_{12} and P_{34} can reach shared keys without any further interactions by a Diffie-Hellman Key-Exchange.*

8. *The shared (output) key will be* $\mathsf{key}_i = \mathsf{m}_{34}^{\mathrm{r}_{12}} = \mathsf{m}_{12}^{\mathrm{r}_{34}} := g^{\hat{s}\hat{s}'}$.

Theorem 10. *For any generic adversary $\mathcal{A}$ against the 4-party NIKE of Construction 9, which places q queries in total to either H or to the group operations (as in Shoup's GGM), the probability that $\mathcal{A}$ outputs the right key is*

$$O(q^2/\lambda^4 + q/\lambda^2 + \mathsf{polylog}(\lambda)/\lambda).$$

Before we proceed with the proof, we make two observations. First, we focus on bounding the probability that $\mathcal{A}$ *finds* the key, while K-NIKE requires that $\mathcal{A}$ cannot distinguish the key from random. However, given the shared key, all four parties can use the standard Goldreich-Levin theorem to extract a hardcore bit of the key, such that *guessing* this shared bit is as hard as *finding* the initial shared key. Hence, Theorem 10 actually implies the security of this modified 4-NIKE protocol. Second, both the construction and the adversary are allowed to query H and the generic group. However, Shoup's generic group model implies in particular the existence of an injective random oracle. Therefore, Theorem 10 further implies the existence of a quadratically secure 4-NIKE protocol in the 'bare' generic group model of Shoup.

4.1 Correctness

Let us first clarify why the protocol works correctly with a *single* round of interaction. Based on Theorem 10, after a single round of interaction (i.e. invoking the message generation algorithm once for every party,) all parties will have the following values:

- $\overline{\mathsf{m}}_1 = \left(H\left(\bar{\mathrm{r}}_1^{(1)}\right), g^{\bar{\mathrm{r}}_1}\right)$ and $\overline{\mathsf{m}}_2 = H\left(\bar{\mathrm{r}}_2^{(1)}\right)$ for parties P_1 and P_2.
- $\overline{\mathsf{m}}_3 = \left(H\left(\bar{\mathrm{r}}_3^{(1)}\right), g^{\bar{\mathrm{r}}_3}\right)$ and $\overline{\mathsf{m}}_4 = H\left(\bar{\mathrm{r}}_4^{(1)}\right)$ for parties P_3 and P_4.

Therefore, with the help of the baby-step giant-step or Pollard's rho algorithm, P_2 and P_4 can find $\hat{s} \subset \bar{\mathrm{r}}_1$ and $\hat{s}' \subset \bar{\mathrm{r}}_3$, having $g^{\bar{\mathrm{r}}_1}$ and $g^{\bar{\mathrm{r}}_3}$ respectively.

Since $H(s) \subset H\left(\bar{\mathrm{r}}_1^{(1)}\right)$ and $H(s') \subset H\left(\bar{\mathrm{r}}_3^{(1)}\right)$ were public from the interaction phase (where s and s' are the first half of $\hat{s}$ and $\hat{s}'$ resp.,) then all parties can find the corresponding values $g^{\hat{s}}$ and $g^{\hat{s}'}$ from $g^{\bar{\mathrm{r}}_1}$ and $g^{\bar{\mathrm{r}}_3}$ respectively. From there, it is easy to see that P_{12} and P_{34} (i.e. all parties,) can find the shared key $g^{\hat{s}\hat{s}'}$ from in steps 7 and 8 of the key generation algorithm.

Now to argue the correctness of the protocol, note that the Msg algorithm has $2 \cdot \lambda\ell(\lambda)$ computations, and the Key algorithm has constant number of computations along with running the Msg and BSGS algorithms. Therefore, the running time of both algorithms are of $\tilde{O}(\lambda)$. Moreover, the Key algorithm aborts and outputs 0 only if either $|H(\bar{\mathrm{r}}_1^{(1)}) \cap H(\bar{\mathrm{r}}_2^{(1)})| = 0$ or $|H(\bar{\mathrm{r}}_3^{(1)}) \cap H(\bar{\mathrm{r}}_4^{(1)})| = 0$, which by birthday paradox (and similar to [BGI08, Theorem 1]'s proof,) only happens with $\mathsf{negl}(\lambda)$ probability.

In order to prove the correctness of the scheme, we need to show that with overwhelming probability, all parties will find the same key. By correctness of the BSGS algorithm, it will follow from the construction that P_1 and P_2, as well as P_3 and P_4, will have the same and correct shared secret key. By correctness of the Diffie-Hellman key-exchange protocol, it follows that P_{12} and P_{34} (i.e. all the four parties) will agree on the same key. □

4.2 Security Analysis

We prove that a generic adversary $\mathcal{A}$ making q queries (either to the injective random oracle or to the group) is able to output the shared key with probability at most $O(q^2/\lambda^4 + q/\lambda^2 + \mathsf{polylog}(\lambda)/\lambda)$. We start by analyzing the following simpler game G:

1. The adversary is given access to a generic group $\mathbb{G}$ of order $p \approx \lambda^4$, and to an injective random oracle H. It can ask q queries in total to either of them.
2. The game samples $(g, g^a, g^b) \xleftarrow{\$} \mathbb{G}^3$ and sends it to the adversary. Let us write $a = a_0||a_1$ and $b = b_0||b_1$, where the a_i, b_i are $2\log(\lambda)$-bit long. The game also computes $(h_a, h_b) \leftarrow (H(a_0), H(b_0))$ and sends it to $\mathcal{A}$.
3. The adversary outputs a group element g^c at the end of the game.

At the end of the game, we say that $\mathcal{A}$ wins if either of the two following conditions is fulfilled:

1. $c = a \cdot b$, or
2. the list L of all queries of $\mathcal{A}$ to H contains either a_0 or b_0.

Claim 11. $\Pr[\mathcal{A}$ *wins in game* $G] \leq q^2/\lambda^4 + 2q/\lambda^4 + 2q/\lambda^2$.

Proof. The proof follows closely to the blueprint of Shoup's proof [Sho97] that the advantage of any q-query adversary against the CDH problem is at most $(q^2 + 2q)/|\mathbb{G}|$. Recall that the generic oracle works as follows: an *encoding function* $\mathsf{enc} : \mathbb{Z}_p \mapsto S$ is defined, where S is a set of bit-string. $\mathcal{A}$ sees group elements as encodings of integers: we identify g^a, g^b with $\mathsf{enc}(a), \mathsf{enc}(b)$. Then, $\mathcal{A}$ can ask *addition queries*, which on input $(\mathsf{enc}(x), \mathsf{enc}(y))$ and a sign $+$ or $-$, outputs $\mathsf{enc}(x \pm y)$. This more "limited" variant of SGGM is equivalent to Definition 3.

Consider the following variant of the game described above. We simulate all accesses to the generic group exactly as in Shoup's proof: we maintain a list of linear bivariate polynomials $(F_1, \cdots, F_k) \in \mathbb{Z}_p[X, Y]$ (initialized with $k = 3$, $F_1 = 1$, $F_2 = X$, and $F_3 = Y$), and a list of distinct values $(\mathsf{enc}_1, \cdots, \mathsf{enc}_k)$ in S (initialized with three random distinct elements of S). For any addition query (i, j), we set $F_{k+1} \leftarrow F_i \pm F_j$. If F_{k+1} matches a polynomial F_t already in the list, let $\mathsf{enc}_{k+1} \leftarrow \mathsf{enc}_t$, otherwise we set it to a uniformly random element in S.

Furthermore, we also simulate all queries to H as follows: we initially sample (h_a, h_b) uniformly at random from $\{0,1\}^\lambda$. We maintain a list L of queries. Each time $\mathcal{A}$ queries i, we search if a tuple (i, h) exists in L; if it does not, we sample h uniformly at random from $\{0,1\}^\lambda$, return h, and add (i, h) to L.

At the end of the game, we sample $(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$ and check if (1) $F_i(x, y) = F_j(x, y)$ for some $i \neq y$, or (2) $F_i(x, y) = xy$ for some i, or (3) writing $x = x_0||x_1$ and $y = y_0||y_1$, the list L contains a pair (x_0, h) or a pair (y_0, h). By the Schwartz-Zippel lemma, the probability for a random (x, y) that (1) or (2) is satisfied is at most $q^2/\lambda^4 + 2q/\lambda^4 = O(q^2/\lambda^4)$. Furthermore, the probability to hit any entry of the L (whose size is at most q) with x_0 or y_0 (which are uniform over $[\lambda^2]$) is at most $2q/\lambda^2$. Overall, the probability that either (1), (2), or (3) holds is at most $q^2/\lambda^4 + 2q/\lambda^4 + 2q/\lambda^2$.

Furthermore, as in [Sho97], we observe that this simulated game differs from the real game exactly when (1), (2), or (3) happens, and that the adversary wins in the real game exactly when this happens: otherwise, the real and simulated game are perfectly indistinguishable. Therefore, the probability that $\mathcal{A}$ wins in the real game is at most $q^2/\lambda^4 + 2q/\lambda^4 + 2q/\lambda^2$; this concludes the proof. □

The rest of the proof proceeds by reducing the existence of an adversary against Construction 9 to the existence of an adversary in the game G. Let $\mathcal{A}$ be a q-query adversary against Construction 9. The reduction proceeds as follow: it receives a challenge $(\mathsf{enc}(a), \mathsf{enc}(b), h_a, h_b)$ from the game G. Then, it samples a random looking transcript of Construction 9, using $\tilde{O}(\lambda)$ queries to the group and to H to generate all group elements and hashes of prefixes, with the following difference: it replaces the lexicographically first collision in the message of P_1 by $\mathsf{enc}(a)$, and sets h_a to be the corresponding prefix hash. It also replaces the corresponding hash collision by h_a in the message of P_2. It does the same with $(\mathsf{enc}(b), h_b)$ with P_3 and P_4.

Observe that this simulated transcript is statistically indistinguishable from an honestly generated transcript of Construction 9: the simulation fails only when the transcript does not contain any hash collision between P_1 and P_2 or between P_3 and P_4. But the probability of not having a collision between $\lambda \cdot \ell(\lambda)$-sized tuples of random elements from $[\lambda^2]$ is negligible by a straightforward Chernoff bound, similar to the proof of Claim 6. Furthermore, by construction, the corresponding shared key for the simulated transcript is exactly $\mathsf{enc}(ab)$.

Therefore, if there exists a q-query adversary against Construction 9 which finds the shared key with probability ε, then there exists a $(q + \tilde{O}(\lambda))$-query adversary against game G which wins the game with probability at least $\varepsilon - \mathsf{negl}(\lambda)$. This concludes the proof. □

5 Impossibility Results

5.1 Defining 3-NIKE in Maurer's Generic Group Model

We first define the model and the problems. Then,we present our results.

Notation. We use bold font to represent random variables.

Definition 12 (Maurer's Generic Group Model (MGGM)). *Let $p \in \mathbb{Z}$ be a positive integer. Let Arr_P be an array for party P initialized to null at all indices except index 1 where it is initialized to be 1. Also, e is the last index of Arr that is not null (so, initially $e = 1$). Parties have access to group elements only through the following operations.*

- Add *query: The party P submits query $\mathsf{Add}(i_1, i_2, c_1, c_2)$ where $i_1, i_2 \in [e]$ and $c_1, c_2 \in \mathbb{Z}_p$. Then, the value $c_1.\mathsf{Arr}_\mathsf{P}[i_1] + c_2.\mathsf{Arr}_\mathsf{P}[i_2]$ will be written at $\mathsf{Arr}_\mathsf{P}[e+1]$ and e will be updated to $e+1$.*
- Equal *query: The party P submits query $\mathsf{Equal}(i_1, i_2)$ where $i_1, i_2 \in [e]$. The party receives 1 if $\mathsf{Arr}_\mathsf{P}[i_1] = \mathsf{Arr}_\mathsf{P}[i_2]$ and 0 otherwise.*
  The following two queries are optional, as they can be obtained from the above (see Remark 13) but we define them as they sometimes help with a better presentation of algorithms in this model.
- Write *query: The party P submits query $\mathsf{Write}(x)$ for a group element $x \in \mathbb{Z}_q$ and then x is written to $\mathsf{Arr}[e+1]$ followed by increasing e by one.*
- Zero *query: The party P submits query $\mathsf{Zero}(c_1, \ldots, c_e)$ where $c_1, \cdots, c_e \in \mathbb{Z}_p$. The party receives 1 if $\sum_{i\in[e]} c_i \mathsf{Arr}_\mathsf{P}[i] = 0$ and 0 otherwise. When $\mathsf{Arr}_\mathsf{P}[1] = 1$, this query can zero-test a general* affine *function over $\mathsf{Arr}_\mathsf{P}[2], \ldots, \mathsf{Arr}_\mathsf{P}[e]$.*

Remark 13 (Comparing queries and the default model). Note that the Equal queries can be simulated using a single call to an Zero query. Hence, Zero queries are as powerful. Conversely, an Zero query can be simulated using e queries to the Add followed by a single query to Equal that compares the result with a prepared encoding of zero. By default, we only allow $\mathsf{Add}, \mathsf{Zero}$ queries, but sometimes we state the availability of Write queries for a clearer presentation.

Definition 14 (3-NIKE in MGGM). *In this model, there are three parties Alice A, Bob B, and Charlie C. All parties receive a security parameter λ, and prime number p and a private randomness as inputs. Let their internal randomness be r_A, r_B, and r_C respectively. Each of the parties has access to a private MGGM oracle, but all parties' groups are defined over the same $\mathbb{Z}_p$. After making queries to the oracle, party P for all $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ will simultaneously perform the following actions:*

- P *sends a string m_P to the other parties. Define transcript tran as $\mathsf{tran} = (\mathsf{m}_\mathsf{A}, \mathsf{m}_\mathsf{B}, \mathsf{m}_\mathsf{C})$. No-message protocols are a special case where $\forall \mathsf{P}, \mathsf{m}_\mathsf{P} = \emptyset$.*
- P *submits query $\mathsf{copy}(\gamma)$. Upon such a query, the last γ indices of P's array will be copied to the end of the other parties' arrays, for some publicly known value $\gamma \leq \mathsf{poly}(\lambda)$ that is* fixed *in the protocol and is the* same *for all three parties. Parties submit this operation at the same time, however, it will be processed first for A, second for B and last for C. Let the vector of group elements that P sends to others be $\overline{\mathrm{q}}_\mathsf{P} = (\mathrm{q}_{\mathsf{P},1}, \cdots, \mathrm{q}_{\mathsf{P},\gamma})$. Additionally, let $\overline{\mathrm{q}}_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}} = (\overline{\mathrm{q}}_\mathsf{A}, \overline{\mathrm{q}}_\mathsf{B}, \overline{\mathrm{q}}_\mathsf{C})$ in which if we run party P with internal randomness r_P, they will copy $\overline{\mathrm{q}}_\mathsf{P}$.*

Without loss of generality (by Lemma 17), the parties in the steps above only use Write queries to their oracle, as they will know the content of their arrays fully.

In the second step, based on their private randomness r_P, transcript tran, and their updated private oracles continue to interact with their MGGM oracles and will write a group element key_P to their local MGGM oracle.[7] *Moreover, we ask that all the parties are efficient (so, they submit at most $\mathsf{poly}(\lambda)$ queries to the oracle). We enforce this by asking all parties to make a copy query with a fixed publicly known parameter $\gamma \leq \mathsf{poly}(\lambda)$ in their first step, and then (after receiving the exchanged messages) ask exactly α Add queries, β Zero queries for publicly known and fixed values of $\alpha, \beta, \leq \mathsf{poly}(\lambda)$.*

Completeness: *We say that parties agree on a key with probability $1 - \delta$ (where δ is called completeness error) if $\Pr[\mathsf{key}_\mathsf{A} = \mathsf{key}_\mathsf{B} = \mathsf{key}_\mathsf{C}] \geq 1 - \delta$, where the probability is over the randomness of the parties. We say that the protocol has perfect completeness if $\delta = 0$.*

Soundness: *We say that Eve E breaks the protocol with advantage ρ, if she finds the key with probability at least than $1/p + \rho$ in the following game. E will get the transcript tran and has access to a private oracle Arr_E that gets as input the result of the copy operations of the* all *three parties. Namely, E's MGGM oracle will contain 1 followed by 3γ group elements that are communicated by the parties in a canonical order. The scheme is secure if the advantage of any $\mathsf{poly}(\lambda)$-time E is at most $\mathsf{negl}(\lambda)$.*

[7] This writing could be due to a direct write operation or deriving the group element from other array elements (in which case the parties do not actually have direct access to the group element itself). However, note that if the group elements are eventually encoded, a la Shoup's model, the encoding of the group element will be accessible to the parties.

Remark 15. The Diffie-Hellman protocol is a 2-party protocol that can be stated in the MGGM with perfect completeness, no (text) messages (in addition to the exchanged group elements), no zero-test queries are asked ($\beta = 0$) and only one group element is sent by each party ($\gamma = 1$).

Remark 16 (Why agreeing on group element in the oracle?). Here we further justify why the key in 3-NIKE protocols in MGGM are written in the $\mathsf{Arr_P}$ oracle, while the oracle's content is not directly accessible to the parties. Firstly, note that MGGM is an idealized model with an oracle Arr which will be eventually substituted with actual encodings (like Shoup's model). Therefore, if an MGGM protocol leads party to an agreement on the same group element x written in their oracles, this will imply an actual agreement when the oracle is substituted with actual encodings. Furthermore, if we ask the parties to agree on a *string* (as the key) in the MGGM, then (1) without loss of generality we can convert the protocol to agreeing on a key $\mathsf{key} \in \mathbb{Z}_p$, one can always bootstrap even a binary key from $\{0,1\}$ to a key from $\{0,1\}^\lambda$, and then round it to a secure key in $\mathbb{Z}_p$, and then (2) the parties can write the secure key $\mathsf{key} \in \mathbb{Z}_p$ in their corresponding oracle without losing any security, in which case they end up agreeing on a secure key as defined in Definition 14.

5.2 Breaking 3-NIKE in the MGGM without zero-test queries

Lemma 17 (Group Elements' Structure in MGGM). *Consider a party* P *who interacts with an MGGM oracle. Suppose* P *receives an input* input *and when it starts to interact with its oracle, there are already* γ *values written in* $\mathsf{Arr_P}$ *(i.e.,* $e = \gamma$ *at the beginning).*[8] *For a fixed* $k > \gamma$*, suppose the algorithm* P *has just written (directly or through an* Add *query) in* $\mathsf{Arr_P}[k]$ *(i.e.,* e *has become* k*), while we have the answer to all of its previous* $\mathcal{A}\mathsf{ff}$ *queries encoded in a vector* vec*. Then, for any such fixed choices of* $\mathsf{input}, k, \mathsf{vec}$*, there are constants* $\mathsf{f}_1, \ldots, \mathsf{f}_\gamma \in \mathbb{Z}_p$*, such that*

$$\mathsf{Arr_P}[k] = \sum_{i \in [\gamma]} \mathsf{f}_i \cdot \mathsf{Arr_P}[i].$$

In particular, if no $\mathcal{A}\mathsf{ff}$ *queries are asked, then for any fixed* input, k*, the value of* $\mathsf{Arr_P}[k]$ *is a linear function of the* γ *group elements that are written in its oracle at the beginning.*

Proof. This observation is used in previous papers (e.g., [FKL17]). In particular, the proof follows by a straightforward induction over k. □

Definition 18 (Compatibility in MGGM). *For any* $\mathcal{S} = \{s_1, \cdots, s_n\} \subseteq \{\mathsf{r_P}, \mathsf{m_P}, \overline{\mathsf{q}}_\mathsf{P}\}_{\mathsf{P} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\}}$ *in a 3-NIKE problem, we say they are compatible if there are internal randomness* $\mathrm{r_A}$*,* $\mathrm{r_B}$*, and* $\mathrm{r_C}$*, using which if we run the protocol, then all* $s_i \in S$ *appears in the protocol. For example we say* $\overline{\mathsf{q}}_\mathsf{A}$*,* $\mathsf{m_A}$*, and* $\mathsf{m_B}$ *are compatible if there exists* $\mathrm{r_A}$*,* $\mathrm{r_B}$*, and* $\mathrm{r_C}$ *using which if we run the protocol we have* $\overline{\mathbf{q}}_\mathsf{A} = \overline{\mathsf{q}}_\mathsf{A}$*,* $\mathbf{m_A} = \mathsf{m_A}$*, and* $\mathbf{m_B} = \mathsf{m_B}$*.*

Lemma 19 (Randomness Switching). *Consider a 3-NIKE protocol with no equality queries. Then for any* $\mathsf{tran} \leftarrow \mathbf{trans}$*,* $\mathrm{r_A}, \mathrm{r'_A} \leftarrow \mathbf{r_A}|\mathsf{tran}$*,* $\mathrm{r_B} \leftarrow \mathbf{r_B}|\mathsf{tran}$*, and* $\mathrm{r_C}, \mathrm{r'_C} \leftarrow \mathbf{r_C}|\mathsf{tran}$ *following holds:*

$$\begin{aligned} &\mathsf{Key_B}(\mathrm{r_B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r_A},\mathrm{r_B},\mathrm{r_C}}) + \mathsf{Key_B}(\mathrm{r_B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r'_A},\mathrm{r_B},\mathrm{r'_C}}) \\ &\qquad = \mathsf{Key_B}(\mathrm{r_B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r_A},\mathrm{r_B},\mathrm{r'_C}}) + \mathsf{Key_B}(\mathrm{r_B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r'_A},\mathrm{r_B},\mathrm{r_C}}) \end{aligned} \tag{7}$$

Additionally, with probability $1 - 4\delta$ *over the randomness of* $\mathsf{tran} \leftarrow \mathbf{trans}$*,* $\mathrm{r_A}, \mathrm{r'_A} \leftarrow \mathbf{r_A}|\mathsf{tran}$*,* $\mathrm{r_B} \leftarrow \mathbf{r_B}|\mathsf{tran}$*, and* $\mathrm{r_C}, \mathrm{r'_C} \leftarrow \mathbf{r_C}|\mathsf{tran}$*, following holds:*

$$\begin{aligned} &\mathsf{Key_A}(\mathrm{r_A}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r_A},\mathrm{r_B},\mathrm{r_C}}) + \mathsf{Key_A}(\mathrm{r'_A}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r'_A},\mathrm{r_B},\mathrm{r'_C}}) \\ &\qquad = \mathsf{Key_C}(\mathrm{r'_C}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r_A},\mathrm{r_B},\mathrm{r'_C}}) + \mathsf{Key_A}(\mathrm{r'_A}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r'_A},\mathrm{r_B},\mathrm{r_C}}) \end{aligned} \tag{8}$$

Proof. By Lemma 17, the final key of Bob is of the following form:

$$\mathsf{key_B} = \mathsf{Key_B}(\mathrm{r_B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r_A},\mathrm{r_B},\mathrm{r_C}}) = \mathsf{f}_{\mathsf{B},\mathsf{B}}(\mathrm{r_B}, \mathsf{tran}) + \textstyle\sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{A},i}(\mathrm{r_B}, \mathsf{tran}) \mathrm{q}_{\mathsf{A},i} + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{C},i}(\mathrm{r_B}, \mathsf{tran}) \mathrm{q}_{\mathsf{C},i}$$

[8] One special case is that $\mathsf{Arr_P}[1] = 1$, but this is not necessary in this Lemma 17.

Therefore we have:

$$\mathsf{Key}_\mathsf{B}(\mathrm{r}_\mathsf{B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}}) + \mathsf{Key}_\mathsf{B}(\mathrm{r}_\mathsf{B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}'_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}'_\mathsf{C}}) =$$

$$(\mathsf{f}_{\mathsf{B},\mathsf{B}}(\mathrm{r}_\mathsf{B}, \mathsf{tran}) + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{A},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}_{\mathsf{A},i} + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{C},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}_{\mathsf{C},i}) +$$

$$(\mathsf{f}_{\mathsf{B},\mathsf{B}}(\mathrm{r}_\mathsf{B}, \mathsf{tran}) + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{A},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}'_{\mathsf{A},i} + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{C},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}'_{\mathsf{C},i}) =$$

$$(\mathsf{f}_{\mathsf{B},\mathsf{B}}(\mathrm{r}_\mathsf{B}, \mathsf{tran}) + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{A},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}'_{\mathsf{A},i} + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{C},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}_{\mathsf{C},i}) +$$

$$(\mathsf{f}_{\mathsf{B},\mathsf{B}}(\mathrm{r}_\mathsf{B}, \mathsf{tran}) + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{A},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}_{\mathsf{A},i} + \sum_{i=1}^{d} \mathsf{f}_{\mathsf{B},\mathsf{C},i}(\mathrm{r}_\mathsf{B}, \mathsf{tran})\mathrm{q}'_{\mathsf{C},i}) =$$

$$\mathsf{Key}_\mathsf{B}(\mathrm{r}_\mathsf{B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}'_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}}) + \mathsf{Key}_\mathsf{B}(\mathrm{r}_\mathsf{B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}'_\mathsf{C}})$$

This proves Equation 7.

We say event $E_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}}$ holds if Alice, Bob and Charlie when executed using randomness r_A , r_B, and r_C, agree on a key. namely:

$$\mathsf{Key}_\mathsf{A}(\mathrm{r}_\mathsf{A}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}}) = \mathsf{Key}_\mathsf{B}(\mathrm{r}_\mathsf{B}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}}) = \mathsf{Key}_\mathsf{C}(\mathrm{r}_\mathsf{C}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}})$$

Note that by reverse sampling, the marginal distribution of $(\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C}, \mathsf{tran})$ is the same as the marginal distribution of $(\mathrm{r}'_\mathsf{A}, \mathrm{r}'_\mathsf{B}, \mathrm{r}'_\mathsf{C}, \mathsf{tran}')$, where $(\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C}) \leftarrow (\mathbf{r}_\mathsf{A}, \mathbf{r}_\mathsf{B}, \mathbf{r}_\mathsf{C})$, tran is the transcript when we run the protocol with r_A, r_B, r_C, $\mathsf{tran}' \leftarrow \mathbf{trans}$, and $(\mathrm{r}'_\mathsf{A}, \mathrm{r}'_\mathsf{B}, \mathrm{r}'_\mathsf{C}) \leftarrow (\mathbf{r}_\mathsf{A}|\mathsf{tran}', \mathbf{r}_\mathsf{B}|\mathsf{tran}', \mathbf{r}_\mathsf{C}|\mathsf{tran}')$. Thus any r_A, r_B, and r_C where $\mathsf{tran} \leftarrow \mathbf{trans}$, $\mathrm{r}_\mathsf{A} \leftarrow \mathbf{r}_\mathsf{A}|\mathsf{tran}$, $\mathrm{r}_\mathsf{B} \leftarrow \mathbf{r}_\mathsf{B}|\mathsf{tran}$, and $\mathrm{r}_\mathsf{C} \leftarrow \mathbf{r}_\mathsf{C}|\mathsf{tran}$, agree on a key with probability $1 - \delta$. Let $E^* = E_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}} \wedge E_{\mathrm{r}'_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}} \wedge E_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}'_\mathsf{C}} \wedge E_{\mathrm{r}'_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}'_\mathsf{C}}$, then by a union bound over the complements of the events on the right hand side of the equation, we have:

$$\Pr_{\mathsf{tran}\leftarrow\mathbf{trans},\mathrm{r}_\mathsf{A},\mathrm{r}'_\mathsf{A}\leftarrow\mathbf{r}_\mathsf{A}|\mathsf{tran},\mathrm{r}_\mathsf{B}\leftarrow\mathbf{r}_\mathsf{B}|\mathsf{tran},\mathrm{r}_\mathsf{C},\mathrm{r}'_\mathsf{C}\leftarrow\mathbf{r}_\mathsf{C}|\mathsf{tran}}[E^*] \geq 1 - 4\delta \tag{9}$$

We finally use Equation 9 to conclude the proof of Equation 8 by switching the corresponding keys in Equation 7.

□

Theorem 20 (Breaking 3-NIKE protocols without zero-test queries). *Suppose Π is a 3-NIKE protocol in the MGGM with no* Equal *or* Zero *queries (i.e., $\beta = 0$) and completeness error δ. Then, there is an adversary Eve who, given the transcript* tran *and oracle access to the 3γ broadcast group elements finds Alice's key with probability $1 - 4\delta$ by asking $O(\alpha)$ queries to its* Add *oracle.*

Proof. Note that in a 3-NIKE in MGGM, by sampling r_P, $\{\overline{\mathrm{q}}_\mathsf{P}, \mathsf{m}_\mathsf{P}\}$ will also be sampled, and m_P only depends on r_P; namely, we can sample a randomness r_P such that $(\mathrm{r}_\mathsf{P}, \mathsf{m}_\mathsf{P})$ is compatible without any query to the MGGM operators. Thus in Lemma 19, given a transcript tran, we can sample any r_P with out any additional queries to the oracle.

Now to break the original protocol where Alice, Bob, and Charlie's respective internal randomness are r_A, r_B, and r_C, and their respective set of copied group elements and messages are $(\overline{\mathrm{q}}_\mathsf{A}, \mathsf{m}_\mathsf{A})$, $(\overline{\mathrm{q}}_\mathsf{B}, \mathsf{m}_\mathsf{B})$, and $(\overline{\mathrm{q}}_\mathsf{C}, \mathsf{m}_\mathsf{C})$, consider the following attack:

Eve samples a new Alice $\mathrm{r}'_\mathsf{A} \leftarrow (\mathbf{r}_\mathsf{A}|\mathsf{tran})$ and a new Charlie $\mathrm{r}'_\mathsf{C} \leftarrow (\mathbf{r}_\mathsf{C}|\mathsf{tran})$, and computes their respective vector of copied group elements $\overline{\mathrm{q}}'_\mathsf{A}$, and $\overline{\mathrm{q}}'_\mathsf{C}$ by running their algorithms. Then Eve finds key_A as follows:

$$\mathsf{key}_\mathsf{E} = \mathsf{Key}_E(\mathrm{r}'_\mathsf{A}, \mathrm{r}'_\mathsf{C}, \overline{\mathrm{q}}_\mathsf{A}, \overline{\mathrm{q}}_\mathsf{B}, \overline{\mathrm{q}}_\mathsf{C}, \overline{\mathrm{q}}'_\mathsf{A}, \overline{\mathrm{q}}'_\mathsf{C}, \mathsf{tran}) =$$

$$\mathsf{Key}_\mathsf{C}(\mathrm{r}'_\mathsf{C}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}'_\mathsf{C}}) + \mathsf{Key}_\mathsf{A}(\mathrm{r}'_\mathsf{A}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}'_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}_\mathsf{C}}) - \mathsf{Key}_\mathsf{A}(\mathrm{r}'_\mathsf{A}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}'_\mathsf{A},\mathrm{r}_\mathsf{B},\mathrm{r}'_\mathsf{C}})$$

First note that Eve needs only $O(\alpha)$ queries to calculate the above formula. To prove that this is the actual key, first note that by reverse sampling the distribution of $(\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C}) \leftarrow (\mathbf{r}_\mathsf{A}, \mathbf{r}_\mathsf{B}, \mathbf{r}_\mathsf{C})$ is the same as the distribution of $(\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C}) \leftarrow (\mathbf{r}_\mathsf{A}|\mathsf{tran}, \mathbf{r}_\mathsf{B}|\mathsf{tran}, \mathbf{r}_\mathsf{C}|\mathsf{tran})$ where $\mathsf{tran} \leftarrow \mathbf{trans}$. Thus we can

w.l.o.g. assume that first the transcript was sampled and then Alice, Bob, and Charlie were sampled conditioned on the transcript. Now by Lemma 19, the following holds with probability at least $1-4\delta$:

$$\mathsf{key}_{\mathsf{E}} = \mathsf{Key}_{\mathsf{A}}(\mathrm{r}_{\mathsf{A}}, \mathsf{tran}, \overline{\mathrm{q}}_{\mathrm{r}_{\mathsf{A}},\mathrm{r}_{\mathsf{B}},\mathrm{r}_{\mathsf{C}}})$$

So Eve finds the key with probability $1-4\delta$ with $O(\alpha)$ queries. □

5.3 Breaking 3-NIKE in the MGGM with zero-test queries

Theorem 21 (Breaking 3-NIKE protocols with equality queries). *Suppose Π is a 3-NIKE protocol in the MGGM with parameters $\alpha, \beta, \gamma \leq \mathsf{poly}(\lambda)$ and completeness error δ (see Definition 14). Then, there is an adversary who, given the transcript* tran *and oracle access to the 3γ broadcast group elements finds Alice's key with probability $1-4\delta-\delta'$ by asking $\mathsf{poly}(\lambda)$ queries to its MGGM oracle. In particular,* E *will ask an expected number of $O(\gamma\beta/\delta')$* Zero *queries and $O(\alpha)$* Add *queries.*

Corollary: quadratic attack in MGGM. If honest parities ask n parties to their oracle and agree on a key with probability ≥ 0.99, then we have $\alpha, \beta, \gamma \leq n$. In this case, E can choose $\delta' = 0.01$, and so it can find Alice's key with probability at least 0.95 by asking $O(n^2)$ queries in total.

In the rest of this subsection, we prove Theorem 21.

Before presenting the attack, we go over some relevant definitions.

Definition 22 (Notation and notions for the attack). *In the following, let fix γ to be known, and let $\mathcal{A}\mathsf{ff}_\gamma = \mathbb{Z}_p^{\gamma+1}$. We interpret each $f = (a_0, \ldots, a_\gamma) \in \mathcal{A}\mathsf{ff}_\gamma$ as an affine function from $\mathbb{Z}^\gamma$ to $\mathbb{Z}$ that maps $x = (x_1, \ldots, x_\gamma) \in \mathbb{Z}_p^\gamma$ to $f(x) = a_0 + \sum_{i\in[\gamma]a_i} x_i$. Using any such f, we can obtain two linear* constraints*: an equality $f(x) = 0$ and an inequality $f(x) \neq 0$. We represent the former constraint using (f, eql) and the latter as (f, nql). We call* LinCon *a set of linear constraints if* LinCon *contains elements that are of the form (f, c) where $f \in \mathcal{A}\mathsf{ff}_\gamma, c \in \{\mathsf{eql}, \mathsf{nql}\}$. We say x satisfies the linear constraint (f, c), if $f(x) = 0$ for $c = \mathsf{eql}$ and $f(x) \neq 0$ for $c = \mathsf{nql}$. We say that x satisfies a* set LinCon *of linear constraints, if x satisfies* all *of the linear constraints in* LinCon*. For any party, let $\mathsf{LinEq}_{\mathsf{P}} = \{f \mid (f, \mathsf{eql}) \in \mathsf{LinCon}_{\mathsf{P}}\}$ be the set of linear* equality *constraints for party* P*. For two sets $\mathsf{LinEq}_{\mathsf{B}}, \mathsf{LinEq}_{\mathsf{C}} \subseteq \mathcal{A}\mathsf{ff}_\gamma$ interpreted as affine constraints over two different set of variables, we define their* combination $\mathsf{LinEq}_{\mathsf{B,C}} \subset \mathbb{Z}_{2\gamma+1}$ *as the set of all vectors $(a_0, a_{\mathsf{B},1}, \ldots, a_{\mathsf{B},\gamma}, a_{\mathsf{C},1}, \ldots, a_{\mathsf{C},\gamma})$, such that either*

$$(a_0, a_{\mathsf{B},1}, \ldots, a_{\mathsf{B},\gamma})\mathsf{LinEq}_{\mathsf{B}} \wedge (a_{\mathsf{C},1}, \ldots, a_{\mathsf{C},\gamma}) = (0, \ldots, 0)$$

or

$$(a_0, a_{\mathsf{C},1}, \ldots, a_{\mathsf{C},\gamma}) \in \mathsf{LinEq}_{\mathsf{C}} \wedge (a_{\mathsf{B},1}, \ldots, a_{\mathsf{B},\gamma}) = (0, \ldots, 0).$$

For a party P*, a message* m_{P} *(sent by that party), and set of linear constraints* LinCon*, we define $\mathcal{R}_{\mathsf{P}} = \mathcal{R}_{\mathsf{P}}(\mathsf{LinCon}, \mathsf{m}_{\mathsf{P}})$ to be the set of random seeds for party* P *that are compatible with* LinCon *and* m_{P}*; namely, $r \in \mathcal{R}_{\mathsf{P}}$ if by using r,* P *outputs the message* m_{P} *and group elements $x = (x_1, \ldots, x_\gamma)$ (to be sent to other parties) such that x satisfies* LinCon*. For any distribution D over $\mathbb{Z}_p^\gamma$, we call $f \in \mathcal{A}\mathsf{ff}_\gamma$ (interpreted as an equality constraint) ε-heavy for D, if $\Pr_{x\leftarrow D}[f(x) = 0] \geq \varepsilon$. We say that f is ε-heavy for party* P *conditioned on* $(\mathsf{LinCon}, \mathsf{m}_{\mathsf{P}})$*, if f is ε-heavy for the uniform D that is obtained by sampling $\mathrm{r} \leftarrow \mathcal{R}_{\mathsf{P}}(\mathsf{LinCon}, \mathsf{m}_{\mathsf{P}})$, and obtaining the γ shared group elements generated by party* P *from* r*; namely, if we sample a random seed for the party* P *conditioned on its message* m_{P} *and the linear constraints in* LinCon *over its produced group elements $x = (x_1, \ldots, x_\gamma)$ (to be sent to other parties), then x will satisfy f (as an equality) with probability at least ε. For any set of vectors V of the same dimension, $\mathsf{Span}(V)$ refers to their span using coefficients in $\mathbb{Z}_p$.*

Construction 23 (Attack on protocols with zero tests) *The adversary* E *attacks 3-NIKE protocols with zero test queries as follows.*

- **Inputs to E:** *The adversary has access to* $\mathsf{tran} = (\mathsf{m}_{\mathsf{A}}, \mathsf{m}_{\mathsf{B}}, \mathsf{m}_{\mathsf{C}})$ *and has oracle access to an array that contains $1 + 3\gamma$ group elements: the first one being 1 followed by the 3γ group elements that are broadcast by $\mathsf{A}, \mathsf{B}, \mathsf{C}$. The adversary is also given an input parameter $\varepsilon \in (0, 1)$.*
  The attack has two phases, a learning *phase followed by a* sample *phase.*

– **Learning phase:** *Originally let three sets of linear constraints* $\mathsf{LinCon}_\mathsf{P}, \mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ *to be empty sets, and define* $\mathcal{R}_\mathsf{P}$ *be the corresponding set of random seeds for the party* P *that is compatible with* m_P *and the linear constraints* $\mathsf{LinCon}_\mathsf{P}$ *(see Definition 22).*
Recall $\mathsf{LinEq}_\mathsf{P} = \{f \mid (f, \mathsf{eql}) \in \mathsf{LinCon}_\mathsf{P}\}$. *Then, as long as there is any party* $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ *and any* $f \in \mathcal{A}\mathsf{ff}_\gamma$ *such that (1)* $f \notin \mathsf{Span}(\mathsf{LinEq}_\mathsf{P})$*, and (2)* f *(as a linear equality) is* ε*-heavy for party* P *conditioned on* $(\mathsf{m}_\mathsf{P}, \mathsf{LinCon}_\mathsf{P})$*, then pick (the first lexicographic such)* f *and ask the corresponding* Zero *query from* Arr_E over the group elements shared by P *to find out whether* $f(\overline{\mathrm{q}}_\mathsf{P}) = 0$ *or not. If* $f(\overline{\mathrm{q}}_\mathsf{P}) = 0$*, then add* (f, eql) *to* $\mathsf{LinCon}_\mathsf{P}$*, and add* (f, nql) *to* $\mathsf{LinCon}_\mathsf{P}$ *otherwise. Proceed to the next phase when no* ε*-heavy remains.*
– **Sampling phase:** *For the 3-NIKE protocol* $(\mathsf{A}', \mathsf{B}', \mathsf{C}')$ *defined below (in which* no *affine queries are asked) use the attack of Theorem 20 to find a key* $\mathsf{key}_{\mathsf{A}'}$ *for* A' *and output that key. For the fixed values of* $(\mathsf{LinCon}_\mathsf{P}, \mathsf{m}_\mathsf{P}), \mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, $\mathsf{A}', \mathsf{B}', \mathsf{C}'$*, work as follows.*
  1. *Party* $\mathsf{P} \in \{\mathsf{A}', \mathsf{B}', \mathsf{C}'\}$ *uniformly will pick* $\mathrm{r}_\mathsf{P} \in \mathcal{R}(\mathsf{m}_\mathsf{P}, \mathsf{LinCon}_\mathsf{P})$*. Then,* P *will send the corresponding message* m_P *and group elements* $\overline{\mathrm{q}}_\mathsf{P} = (\mathrm{q}_{\mathsf{P},1}, \cdots, \mathrm{q}_{\mathsf{P},\gamma})$ *that are uniquely produced using* r_P*. Note that the messages of the parties will remain the same as the one fixed in the previous phase (e.g.,* $\mathsf{m}_{\mathsf{A}'} = \mathsf{m}_\mathsf{A}$*), but their broadcast group elements might change.*
  We explain the next step only for A'*; algorithms for* B', C' *are similar.*
  2. *Party* A' *will run the same algorithm as* A *using* $\mathrm{r}_{\mathsf{A}'}$*, but when it comes to any* Zero *query* t*,* P' *will* not *ask it from its oracle and instead will do the following. By Lemma 17, any* Zero *query* t *by* A' *is equivalent to asking a query* $t' = (t_0, t_{\mathsf{B}',1}, \ldots, t_{\mathsf{B}',\gamma}, t_{\mathsf{C}',1}, \ldots, t_{\mathsf{C}',\gamma}) \in \mathbb{Z}_{2\gamma+1}$ *over the* 2γ *group elements* $\overline{\mathrm{q}}_{\mathsf{B}'}, \overline{\mathrm{q}}_{\mathsf{C}'}$ *that are copied to the array of* A' *by parties* B', C'*. Informally speaking,* t' *will be answered* 1 *if and only if this can be concluded from the* equality *constraints for* B, C*. More formally, let* $\mathsf{LinEq}_{\mathsf{B},\mathsf{C}} \subset \mathbb{Z}_{2\gamma+1}$ *be the combination of* $\mathsf{LinEq}_\mathsf{B}, \mathsf{LinEq}_\mathsf{C}$ *as in Definition 22, and answer the* Zero *query* t' *by* 1 *if and only if* $t' \in \mathsf{Span}(\mathsf{LinEq}_{\mathsf{B},\mathsf{C}})$*. If* $t' \in \mathsf{Span}(\mathsf{LinEq}_{\mathsf{B},\mathsf{C}})$*, we call* t' *a trivial query. After emulating* A*,* A' *will output the key* $\mathsf{key}_{\mathsf{A}'}$ *that* A *would output.*

Lemma 24 (Efficiency). *The expected number of zero-test queries asked by* E *in Construction 23 is* $\leq 3\gamma/\varepsilon$*.*

Proof. We prove that the expected number of the queries that E asks for each party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ is at most γ/ε. Then, the lemma follows from the linearity of expectation. Now for a party P let $f_1, f_2, \cdots$ be the sequence of the queries that E asks, and if a query is not asked we let it be $\perp$. Let $p_i = \Pr[f_i \neq \perp]$. Let t be a random variable of the number of the zero-test queries that E ask over $\overline{\mathrm{q}}_\mathsf{P}$, then $\mathbb{E}[t] = \sum p_i$. Additionally define ZT to be the set of *all* zero-tests that pass over $\overline{\mathrm{q}}_\mathsf{P}$. Note that ZT is a random variable determined by the randomness of Alice. Moreover, define random variables $S_i = \mathsf{Span}(f_j \mid f_j \in \mathsf{ZT} \text{ for } j \leq i)$ $d_i = \dim(S_i)$. Note that as $\dim(\mathsf{ZT}) \leq \gamma$, $d_i \leq \gamma$ for all i. Now we claim that

$$\mathbb{E}[d_i] - \mathbb{E}[d_{i-1}] \geq p_i \cdot \varepsilon. \tag{10}$$

Since d_i is either d_{i-1} or $d_{i-1} + 1$, we have $\mathbb{E}[d_i] - \mathbb{E}[d_{i-1}] = \mathbb{E}[d_i - d_{i-1}] = \Pr[d_i = d_{i-1} + 1]$. By the definition, if $f_i \neq \perp$, then $f_i \notin S_{i-1}$, thus we have:

$$\Pr[d_i = d_{i-1} + 1] = \mathbb{E}_{f_1,\ldots,f_{i-1}} \Pr[f_i \neq \perp \wedge f_i \in \mathsf{ZT} \mid f_1, \ldots, f_{i-1}] =$$

$$\Pr[f_i \neq \perp \mid f_1, \ldots, f_{i-1}] \cdot \Pr[f_i \in \mathsf{ZT} \mid f_1, \ldots, f_{i-1}, f_i \neq \perp].$$

By the definition it holds that $\Pr[f_i \in \mathsf{ZT} \mid f_1, \ldots, f_{i-1}, f_i \neq \perp] \geq \varepsilon$, and so

$$\Pr[d_i = d_{i-1} + 1] \geq \varepsilon \cdot \mathbb{E}_{f_1,\ldots,f_{i-1}} \Pr[f_i \neq \perp \mid f_1, \ldots, f_{i-1}] = \varepsilon \cdot p_i.$$

So we have proved the claim of Equation 10. Note that the total number of linear constraints over γ variables is $p^{\gamma+1}$. Thus using Equation 10 we have:

$$\sum_{i=q}^{p^{\gamma+1}} \mathbb{E}[d_i] \geq \sum_{i=1}^{p^{\gamma+1}} \mathbb{E}[d_{i-1}] + \sum_{i=0}^{p^{\gamma+1}} p_i.\varepsilon.$$

$$\rightarrow \gamma \geq \mathbb{E}[d_{p^{\gamma+1}}] \geq \sum_{i=0}^{p^{\gamma+1}} p_i.\varepsilon.$$

$$\rightarrow \gamma/\varepsilon \geq \mathbb{E}[d_{p^{\gamma+1}}] \geq \sum_{i=0}^{p^{\gamma+1}} p_i = \mathbb{E}[t].$$

This concludes the proof. □

We will prove the following technical lemma, which will be useful for proving the success probability of the adversary of Construction 23.

Lemma 25 (Heaviness of pure vs. impure constraints). *Suppose we have*

1. *D_1 and D_2 are two distributions over $\mathbb{Z}_p^\gamma$.*
2. *$D_{1,2}$ is the distribution over $\mathbb{Z}_p^{2\gamma}$ that is obtained by independently sampling $\overline{x} = (x_1, \cdots, x_\gamma) \leftarrow D_1, \overline{y} = (y_1, \cdots, y_\gamma) \leftarrow D_2$ and outputting the vector $(\overline{x}, \overline{y})$ of dimension 2γ.*
3. *$\mathsf{LinEq}_1, \mathsf{LinEq}_2$ are subsets of $\mathcal{A}\mathsf{ff}_\gamma = \mathbb{Z}_p^{\gamma+1}$.*
4. *If $\mathsf{P} \in \{1, 2\}$ and $f \in \mathcal{A}\mathsf{ff}_\gamma$ is ε-heavy for D_P, then $f \in \mathsf{Span}(\mathsf{LinEq}_\mathsf{P})$.*

Then, for every $f \in \mathcal{A}\mathsf{ff}_{2\gamma}$ that is ε-heavy for $D_{1,2}$, it holds that $f \in \mathsf{Span}(\mathsf{LinEq}_{1,2})$, where $\mathsf{LinEq}_{1,2}$ is the combination of $\mathsf{LinEq}_1, \mathsf{LinEq}_2$ as in Definition 22.

Proof. Let $f = (a_0, a_{1,1}, \cdots, a_{1,\gamma}, a_{2,1}, \cdots, a_{2,\gamma})$ be ε-heavy for $D_{1,2}$. For $j \in [p]$ define $\varepsilon_{1,j}$, and $\varepsilon_{2,j}$ as follows:

$$\Pr_{x \leftarrow D_1}\left[\sum_{i=1}^{\gamma} a_{1,i}x_i + j = 0 \mod p\right] = \varepsilon_{1,j},$$

$$\Pr_{y \leftarrow D_2}\left[\sum_{i=1}^{\gamma} a_{2t,i}x_i + a_0 - j = 0 \mod p\right] = \varepsilon_{2,j}.$$

Therefore, we have

$$\sum_{j=1}^{p} \varepsilon_{1,j}.\varepsilon_{2,j} = \Pr[f(\overline{x}, \overline{y}) = 0] \geq \varepsilon.$$

Because $\sum_j \varepsilon_{1,j} = \sum_j \varepsilon_{2,j} = 1$, there are $j_1, j_2 \in [p]$ such that $\varepsilon_{1,j_1}, \varepsilon_{2,j_2} \geq \varepsilon$, and so $f_1 = (j_1, a_{1,1}, \cdots, a_{1,\gamma})$ is ε-heavy for D_1 and $f_2 = (a_0 - j_2, a_{2,1}, \cdots, a_{2,\gamma})$ is ε-heavy for D_2. Therefore, by Item 4, $f_1 \in \mathsf{Span}(\mathsf{LinEq}_1)$ and $f_2 \in \mathsf{Span}(\mathsf{LinEq}_2)$, which means $\varepsilon_{1,j_1} = \varepsilon_{2,j_2} = 1$. Furthermore, $\sum_{j=1}^{p} \varepsilon_{1,j} \cdot \varepsilon_{2,j}$ is 1 if $j_1 = j_2$ and is 0 otherwise. Since $\sum_{j=1}^{p} \varepsilon_{1,j}.\varepsilon_{2,j} \geq \varepsilon$, then $j_1 = j_2$. This means that $f \in \mathsf{Span}(\mathsf{LinEq}_{1,2})$. □

For the next two lemmas, let $\mathsf{LinCon} = (\mathsf{LinCon}_\mathsf{P})_{\mathsf{P} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\}}$ be the set of linear constraints discovered by E at the end of the learning phase in Construction 23.

Lemma 26 (Independence of random seeds). *For every fixed $\mathsf{tran}, \mathsf{LinCon}$ at the end of learning phase, the following two distributions are the same:*

- *The joint distribution over the randomness of $\mathsf{A}, \mathsf{B}, \mathsf{C}$ conditioned on being compatible with $\mathsf{tran}, \mathsf{LinCon}$.*
- *Independently sampling randomness of each party P conditioned on being compatible with $(\mathsf{m}_\mathsf{P}, \mathsf{LinCon}_\mathsf{P})$ (and putting them together).*

Proof. The proof is similar to the observation that parties' randomness in interactive protocols, conditioned on the transcript, is always a product distribution.

If $(\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C})$ is compatible with $\mathsf{tran}, \mathsf{LinCon}$, then clearly r_P is compatible with $\mathsf{m}_\mathsf{P}, \mathsf{LinCon}_\mathsf{P}$ for all $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ as well. The more interesting observation is the reverse: if r_P is compatible with $\mathsf{m}_\mathsf{P}, \mathsf{LinCon}_\mathsf{P}$ for all $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, then $(\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C})$ is compatible with $\mathsf{tran}, \mathsf{LinCon}$. That is because, these local compatibilities will guarantee that the protocol will proceed consistently as a whole. □

Lemma 27 (Statistical closeness of two protocols). *For every $\mathsf{tran}, \mathsf{LinCon}$ at the end of learning phase, sample $\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C}$, while $\mathrm{r}_\mathsf{P}, \mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ is sampled (only) conditioned on being compatible with $\mathsf{m}_\mathsf{P}, \mathsf{LinCon}_\mathsf{P}$. Then, do as follows.*

1. *Run the protocol Π (with zero-test queries) using the random seeds $\mathrm{r}_\mathsf{A}, \mathrm{r}_\mathsf{B}, \mathrm{r}_\mathsf{C}$ and output the keys that the parties generate $(\mathsf{key}_\mathsf{A}, \mathsf{key}_\mathsf{B}, \mathsf{key}_\mathsf{C})$.*

2. *Run the protocol* Π' (without *zero-test queries) using the random seeds* $\mathsf{r_A}, \mathsf{r_B}, \mathsf{r_C}$ *and output the keys that parties generate* $(\mathsf{key_{A'}}, \mathsf{key_{B'}}, \mathsf{key_{C'}})$.

Then, for every fixed $\mathsf{tran}, \mathsf{LinCon}$ *at the end of learning phase, with probability at least* $1 - 3\beta \cdot \varepsilon$ *over the randomness of sampling the random seeds, it holds that* $\mathsf{key_P} = \mathsf{key_{P'}}$ *for* all $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ *(simultaneously).*

Proof. For the same set of $\mathsf{tran}, \mathsf{LinCon}$ and random seeds $\mathsf{r_A}, \mathsf{r_B}, \mathsf{r_C}$ the two protocols Π, Π' produce the same keys for the same parties if all the non-trivial zero-test queries are answered negatively.

We now prove that this is indeed the case, by proving that the two games will proceed the same with probability $1 - 3\beta \cdot \varepsilon$.

We define $3\beta + 1$ games $\Pi = \Pi_0, \ldots, \Pi_{3\beta} = \Pi'$ as follows. Let us imagine running A first, then B, and then C. This means that we have a total order on the 3β queries asked by them. In Π_i, the first i zero-test queries are answered similarly to Π' (i.e., using canonical answers) and the rest of them are answered similarly to how Π does answer them.

Let E_i be the event, defined in both Π_i and Π_{i-1}, that the answer to the ith query is not trivially yes, but it is indeed answered to be yes if asked from the real oracle (as if we are in Π_{i-1}). Note that the probability of E_i is the same in both Π_i and Π_{i-1}, and E_i happening is the only way that these experiments differ. All we have to do is to show that the probability of E_i is at most ε.

Suppose $\Pr[E_i] > \varepsilon$, and for simplicity suppose ith query is asked by A. Then, due to the independence of the random $\mathsf{r_A}, \mathsf{r_B}, \mathsf{r_C}$, there is a way to fix $\mathsf{r_A}$ to $\mathsf{r_A^0}$, such that $\Pr[E_i \mid \mathsf{r_A^0}] > 0$. Fixing $\mathsf{r_A^0}$ will fix the coefficients of the ith zero test. This means that there will be an affine test over the group elements shared by B, C that is zero with probability $> \varepsilon$. By Lemma 25, it means that for either of $\mathsf{P} \in \{\mathsf{B}, \mathsf{C}\}$, there is an affine test over the group elements shared by P that is ε heavy, while it is not learned by E, which is a contradiction. □

Finally, we prove Theorem 21 using the lemmas above.

Proof (of Theorem 21). The efficiency of the attacker of Construction 23 follows directly from Lemma 24. So, in the rest of the proof we focus on the success probability of the adversary in guessing Alice's key.

Let $\mathsf{r_A}, \mathsf{r_B}, \mathsf{r_C}$ be the randomness of the parties, tran be the transcript, and $\mathsf{LinCon} = (\mathsf{LinCon_P})_{\mathsf{P} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\}}$ be the result of the learning phase. Define $\delta_{\mathsf{tran},\mathsf{LinCon}}$ to be the completeness error *only conditioned* on $(\mathsf{tran}, \mathsf{LinCon})$. Then, we have $\mathbb{E}_{\mathsf{tran},\mathsf{LinCon}}[\delta_{\mathsf{tran},\mathsf{LinCon}}] = \delta$. We claim that for every fixed $(\mathsf{tran}, \mathsf{LinCon})$, the adversary finds Alice's true key with probability at least $1 - 4(3\beta\varepsilon + \delta_{\mathsf{tran},\mathsf{LinCon}}) - 3\beta\varepsilon$. Below, we prove this. By Lemma 26, sampling $\mathsf{r_A}, \mathsf{r_B}, \mathsf{r_C}$ *jointly* conditioned on $(\mathsf{tran}, \mathsf{LinCon})$ (which itself is equivalent to sampling everything according to the real protocol Π) is equivalent to sampling $\mathsf{r_P}$ independently only conditioned on $(\mathsf{m_P}, \mathsf{LinCon_P})$ for all $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. By Lemma 27, if we use the randomness $\mathsf{r_A}, \mathsf{r_B}, \mathsf{r_C}$ to run protocols Π or Π', with probability at least $1 - 3\beta\varepsilon$, the same set of keys will be produced.

Therefore, conditioned on $(\mathsf{tran}, \mathsf{LinCon})$, the protocol Π' (which is defined based on $\mathsf{tran}, \mathsf{LinCon}$) has completeness error at most $3\beta\varepsilon + \delta_{\mathsf{tran},\mathsf{LinCon}}$. This means that the attacker of Construction 23 will find Alice's key in Π' with probability at least $1 - 4(3\beta\varepsilon + \delta_{\mathsf{tran},\mathsf{LinCon}})$. By another application of Lemma 27, this means that the *same* attacker is finding the true key of Alice (in Π) with probability at least $1 - 4(3\beta\varepsilon + \delta_{\mathsf{tran},\mathsf{LinCon}}) - 3\beta\varepsilon$.

Putting things together, E finds Alice's *true* key with probability at least

$$\mathbb{E}_{\mathsf{tran},\mathsf{LinCon}}[1 - 15\beta\varepsilon - 4\delta_{\mathsf{tran},\mathsf{LinCon}}] = 1 - 15\beta\varepsilon - 4\mathbb{E}_{\mathsf{tran},\mathsf{LinCon}}[\delta_{\mathsf{tran},\mathsf{LinCon}}] = 1 - 15\beta\varepsilon - 4\delta.$$

By choosing $15\beta\varepsilon = \delta'$, the probability of not finding Alice's key will be at most $4\delta + \delta'$, while the expected number of its queries during the learning phase will be $3\gamma/\varepsilon = 45\gamma \cdot \beta/\delta'$. □

References

BC22. C. Brzuska and G. Couteau. On building fine-grained one-way functions from strong average-case hardness. In *EUROCRYPT 2022, Part II*, *LNCS* 13276, pages 584–613. Springer, Heidelberg, May / June 2022. 2, 3

BGI08. E. Biham, Y. J. Goren, and Y. Ishai. Basing weak public-key cryptography on strong one-way functions. In *TCC 2008*, *LNCS* 4948, pages 55–72. Springer, Heidelberg, March 2008. 2, 3, 9, 10, 11, 13

BHK+11. G. Brassard, P. Høyer, K. Kalach, M. Kaplan, S. Laplante, and L. Salvail. Merkle puzzles in a quantum world. In *CRYPTO 2011*, *LNCS* 6841, pages 391–410. Springer, Heidelberg, August 2011. 2

BJK+18. Z. Brakerski, A. Jain, I. Komargodski, A. Passelègue, and D. Wichs. Non-trivial witness encryption and null-iO from standard assumptions. In *SCN 18*, *LNCS* 11035, pages 425–441. Springer, Heidelberg, September 2018. 3

BM09. B. Barak and M. Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In *CRYPTO 2009*, *LNCS* 5677, pages 374–390. Springer, Heidelberg, August 2009. 2

BM17. B. Barak and M. Mahmoody-Ghidary. Merkle's key agreement protocol is optimal: An $O(n^2)$ attack on any key agreement from random oracles. *Journal of Cryptology*, 30(3):699–734, July 2017. 7

BMG07. B. Barak and M. Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 680–688. IEEE, 2007. 7

BRSV17. M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Average-case fine-grained hardness. In *49th ACM STOC*, pages 483–496. ACM Press, June 2017. 2

BRSV18. M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Proofs of work from worst-case assumptions. In *CRYPTO 2018, Part I*, *LNCS* 10991, pages 789–819. Springer, Heidelberg, August 2018. 2

BZ14. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO 2014, Part I*, *LNCS* 8616, pages 480–499. Springer, Heidelberg, August 2014. 2

CG18. M. Campanelli and R. Gennaro. Fine-grained secure computation. In *TCC 2018, Part II*, *LNCS* 11240, pages 66–97. Springer, Heidelberg, November 2018. 2

DH76. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 2

DH21. I. Dinur and B. Hasson. Distributed merkle's puzzles. In *TCC 2021, Part II*, *LNCS* 13043, pages 310–332. Springer, Heidelberg, November 2021. 2

DHH+21. N. Döttling, D. Hartmann, D. Hofheinz, E. Kiltz, S. Schäge, and B. Ursu. On the impossibility of purely algebraic signatures. In *TCC 2021, Part III*, *LNCS* 13044, pages 317–349. Springer, Heidelberg, November 2021. 3

DVV16. A. Degwekar, V. Vaikuntanathan, and P. N. Vasudevan. Fine-grained cryptography. In *CRYPTO 2016, Part III*, *LNCS* 9816, pages 533–562. Springer, Heidelberg, August 2016. 2

EWT21. S. Egashira, Y. Wang, and K. Tanaka. Fine-grained cryptography revisited. *Journal of Cryptology*, 34(3):23, July 2021. 2

FHKP13. E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In *PKC 2013*, *LNCS* 7778, pages 254–271. Springer, Heidelberg, February / March 2013. 2

FKL17. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. Cryptology ePrint Archive, Paper 2017/620, 2017. https://eprint.iacr.org/2017/620. 16

GKRS22. S. Guo, P. Kamath, A. Rosen, and K. Sotiraki. Limits on the efficiency of (ring) LWE-based non-interactive key exchange. *Journal of Cryptology*, 35(1):1, January 2022. 2

Jou00. A. Joux. A one round protocol for tripartite diffie–hellman. In *International algorithmic number theory symposium*, pages 385–393. Springer, 2000. 2

LLW19. R. LaVigne, A. Lincoln, and V. V. Williams. Public-key cryptography in the fine-grained setting. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 605–635. Springer, Heidelberg, August 2019. 2

Mau05. U. M. Maurer. Abstract models of computation in cryptography (invited paper). In *10th IMA International Conference on Cryptography and Coding*, *LNCS* 3796, pages 1–12. Springer, Heidelberg, December 2005. 3

Mer74. R. Merkle. C.s. 244 project proposal. In *Facsimile available at http://www.merkle.com/1974.*, 1974. 2, 9

Mer78. R. C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978. 2, 3

Pol75. J. M. Pollard. A Monte Carlo method for factorization. *BIT*, 15:331–334, 1975. URL: `http://cr.yp.to/bib/entries.html#1975/pollard`. 5, 6

Sha71. D. Shanks. Class number, a theory of factorization, and genera. In *Proc. of Symp. Math. Soc., 1971*, pages 41–440, 1971. 5

Sho97. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT'97*, *LNCS* 1233, pages 256–266. Springer, Heidelberg, May 1997. 3, 14

WP22. Y. Wang and J. Pan. Non-interactive zero-knowledge proofs with fine-grained security. In *EUROCRYPT 2022, Part II*, *LNCS* 13276, pages 305–335. Springer, Heidelberg, May / June 2022. 2

Zha22. M. Zhandry. To label, or not to label (in generic groups). In *CRYPTO'22*, 2022. 3