Fine-Grained Non-Interactive Key Exchange, Revisited

Balthazar Bauer[1], Geoffroy Couteau[2], and Elahe Sadeghi[3]

[1] UVSQ, France
[2] CNRS, IRIF, Université Paris Cité, France
[3] University of Texas at Austin, United States of America
balthazar.bauer2@uvsq.fr, couteau@irif.fr, elahesadeghi@utexas.edu

Abstract. We revisit the construction of multiparty non-interactive key-exchange protocols with fine-grained security, which was recently studied in (Afshar et al., Eurocrypt 2023). Their work introduced a 4-party non-interactive key exchange with quadratic hardness, and proved it secure in Shoup's generic group model. This positive result was complemented with a proof that n-party non-interactive key exchange with superquadratic security cannot exist in Maurer's generic group model, for any $n \geq 3$. Because Shoup's model is stronger than Maurer's model, this leaves a gap between the positive and the negative result, and their work left as an open question the goal of closing this gap, and of obtaining fine-grained non-interactive key exchange without relying on idealized models.

In this work, we make significant progress on both questions. We obtain two main results:

- A 4-party non-interactive key exchange protocol with quadratic security gap, assuming the existence of exponentially secure injective pseudorandom generators, and the subexponential hardness of the computational Diffie-Hellman assumption. In addition, our scheme is conceptually simpler, and can be generalized to other settings (with more parties or from other assumptions).
- Assuming the existence of non-uniformly secure injective pseudorandom generators with exponential hardness, we further show that our protocol is secure in Maurer's model, albeit with a smaller hardness gap (up to $N^{1.6}$), making progress on filling the gap between the positive and the negative result of (Afshar et al., Eurocrypt 2023). Somewhat intriguingly, proving the security of our scheme in Maurer's idealized model turns out to be significantly harder than proving its security in the standard model.

1 Introduction

A non-interactive key exchange (NIKE) allows participants to agree on a common key while hiding the agreed-upon key from external observers. NIKE is one of the most fundamental cryptographic primitives, dating back to the seminal work of Diffie and Hellman [DH76]. In the two-party setting, non-interactive

key exchange is known to exist under a variety of cryptographic assumptions, such as the Diffie-Hellman assumption [DH76], the LWE assumption with super-polynomial modulus-to-noise ratio [GKRS22], and assumptions related to the hardness of factoring [FHKP13].

In contrast, in the *multiparty* setting with $n > 2$ parties, n-NIKE protocols have proven considerably harder to construct. Already in the 3-party setting, the only known construction from standard assumptions relies on the bilinear Diffie-Hellman assumption over pairing groups [Jou00]. For $n > 3$, the only known constructions rely on cryptographic heavy hammers such as indistinguishability obfuscation [BZ14] or multilinear maps [CLT13].

Fine-grained cryptography. The traditional definition of cryptographic primitives requires their security to hold against arbitrary polynomial-time adversaries. Fine-grained cryptography studies the existence of cryptographic primitives when the adversarial power is more restricted (for example, when their runtime must remain below a fixed polynomial bound). Because fine-grained cryptography impose more restrictions on the adversary, it yields variants of standard cryptographic primitives that might be easier to obtain from standard assumptions, while offering meaningful security guarantees.[4] The study of fine-grained cryptography dates back to the seminal paper of Merkle [Mer74, Mer78], which introduced a two-party NIKE using only unstructured hardness (a random oracle) with quadratic hardness gap. In the past few years, it has attracted a significant amount of attention [BGI08, BHK+11, DVV16, BRSV17, BRSV18, CG18, LLW19, EWT21, DH21, WP22, BM09, BC22, ACMS23].

Non-interactive key exchange in the fine-grained setting. In a recent work [ACMS23], motivated by the fact that constructions of multiparty NIKE from standard assumptions have proven elusive so far, Afshar, Couteau, Mahmoody, and Sadeghi (ACMS) initiated the study of multiparty NIKE in the fine-grained setting. The main result of their paper was twofold:

- there exists a 4-party NIKE with quadratic hardness gap over a group Gen which can be proven secure in Shoup's generic group model [Sho97];
- no N-party NIKE with $n \geq 3$ in Maurer's generic group model [MW98]) can achieve super-quadratic security.

Informally, in the generic group model, the parties and the adversary have oracle access to the group operations. In Maurer's model, the group elements are represented as values in an array (stored in the oracle) and the parties cannot see them, but can test the equality between elements using oracle queries. In Shoup's model, a representation of the group elements computed through oracle queries via a random injective mapping is given to the parties (letting them in particular test locally the equality between group elements). Together, these result demonstrate that in contrast with the standard setting of security against

[4] For example, a primitive with a quadratic gap between the runtime of the honest parties and that of the best-possible adversary could be realistically usable: running 2^{40} operations requires a moderate amount of time on a standard computers, while $(2^{40})^2 = 2^{80}$ remains out of reach of anyone but state-level organizations.

all polytime adversaries, 4-NIKE can be constructed in the fine-grained setting (with quadratic hardness gap) using only a generic group. The impossibility result further suggests that without making a non-black-box use of the group, this is the best possible result one can hope for. Nevertheless, the result of ACMS leaves two important questions open:

1. Is it possible to build multiparty NIKE in the fine-grained setting under standard cryptographic assumptions, without relying on idealized models?
2. Is it possible to close the gap between the positive result of ACMS (which holds in Shoup's GGM) and the negative result of ACMS (which holds in Maurer's GGM)?

1.1 Our results

In this work, we answer affirmatively the first of the two questions above, and make significant progress towards answering the second question. Concretely:

1. Assuming the existence of exponentially secure injective pseudorandom generators, and the sub-exponential hardness of CDH, there exists a 4-NIKE with quadratic hardness gap.
2. Assuming the existence of exponentially secure injective pseudorandom generators, there exists a 4-NIKE with sub-quadratic hardness gap in Maurer's generic group model.

Above, *exponentially secure* refers to security against algorithms running in time $2^{c\cdot\lambda}$, where λ is the security parameter, and c is some fixed constant (looking ahead, in our results, we will need c to be $1-\varepsilon$ for a small constant ε — that is, we need near-optimal security of the PRG), and sub-exponential security refers to security against $2^{o(\lambda)}$-time adversaries. Quadratic hardness gap means that if the honest parties run in time n, an adversary running in time $O(n^{2-\varepsilon})$ for an arbitrary constant $\varepsilon > 0$ should have vanishing probability of breaking the protocol.

Our second contribution falls short of fully closing the gap between the positive and negative results of ACMS on two aspects: first, we only prove the existence of a 4-NIKE in Maurer's GGM conditioned on the existence of exponentially secure PRGs. Second, and unlike our first result, our second result crucially requires the existence of a *single* PRG $\texttt{Gen} : \{0,1\}^\lambda \to S$ (as opposed to a family of PRGs) which is exponentially secure against *non-uniform* adversaries. Recent breakthroughs on time-space tradeoffs for inverting functions have shown that such a PRG can always be broken in time $2^{4\lambda/5}$ [MP23, HIW23]. In turn, this implies that the hardness gap of our 4-NIKE can be at most $N^{8/5-\varepsilon}$ for some small $\varepsilon > 0$ (where N denote the runtime of the honest parties). We note that our use of a PRG is not inherent: the property we need is a *combinatorial* property of a mapping from short seeds to a set S, which is related (though not identical) to the construction of Sidon set in additive combinatorics [O'B04]. Unfortunately, explicit constructions of mapping satisfying this combinatorial property remain to our knowledge an open problem in additive combinatorics:

using the best-known construction from [MMN06] yields an unconditional construction of 4-NIKE with hardness gap $N^{1.2-\varepsilon}$ in Maurer's model, far from the optimal $N^{2-\varepsilon}$ hardness gap. Alternatively, using a result of Schnorr [Sch01a] yields an unconditional 4-NIKE with optimal hardness gap, but at the cost of assuming a common reference string of length $\tilde{\Omega}(N^2)$ (larger than the honest parties' runtime) to which the honest parties are given efficient RAM access; we refer the reader to Section 4.4 for more details.

1.2 Technical Overview

As a starting point, let us recall the 4-party NIKE of [ACMS23]. Consider four parties grouped in pairs (Alice, Bob) and (Carole, Deva). Let Gen be a group of prime order p with generator g. Let us denote $\lambda \leftarrow (\log p)/2$ (that is, $p \approx 2^{2\lambda}$) and $N \leftarrow \lambda \cdot 2^\lambda$. The protocol proceeds as follows:

- Alice picks N random elements $(a_1, \cdots, a_N)$ of $\mathbb{Z}_p$, and broadcasts $(g^{a_1}, \cdots, g^{a_N})$. In addition, she interprets each a_i as a pair (a_i^0, a_i^1) of 2λ-bit strings, and broadcasts $(H(a_i^0), \cdots, H(a_i^0))$, where $H(\cdot)$ is a suitable injective hash function.
- Bob picks N random 2λ-bit strings b_i^0 and broadcasts $(H(b_i^0), \cdots, H(b_i^0))$.

The intuition of this first step is the following: the pair (Alice,Bob) will engage in a Diffie-Hellman protocol with the pair (Carole,Deva), to agree on a single shared key. To do so, Alice and Bob must non-interactively agree on a public key and secret key pair, while also revealing the public key to Carole and Deva.

At the heart of this agreement procedure is the birthday paradox: if Alice and Bob could get a *collision* on group elements (i.e., Bob sends some g^{b_j} that collides with one of Alice's g^{a_i}), then Alice and Bob would manage to agree on a pair (a_i, g^{a_i}). Unfortunately, collisions are highly unlikely to happen among group elements: because Alice and Bob run in time $O(N) = \tilde{O}(2^\lambda)$, we want the protocol to resist $o(N^2)$-time adversary, which requires taking a $2^{4\lambda}$-sized group due to square-root-time attacks on the discrete logarithm. Fortunately, Alice and Bob can agree on a *half collision*: by exchanging hashes of the first 2λ bits of Alice's exponents together with N hashes of random 2λ-bit strings from Bob, since $N = \lambda \cdot 2^\lambda$, Alice and Bob are guaranteed to get a collision among hashes with overwhelming probability. This means that for some i, Bob will know *one half* of the exponent of g^{a_i}, namely, a_i^0. Using Pollard's kangaroo algorithm [Pol75], Bob can use this information to compute the discrete logarithm of g^{a_i} in $2^{|a_i^0|/2} \approx 2^\lambda \ll N$ steps.

Following this procedure, Alice and Bob agree on, say, the lexicographically first g^{a_i} whose corresponding $H(a_i^0)$ collides with some $H(b_j^0)$ from Bob. Bob computes the discrete logarithm of g^{a_i}, and both players agree on a joint public key – secret key pair (g^{a_i}, a_i). Then, Carole and Deva execute a similar procedure to agree on a pair (g^{c_i}, c_i). Note that at this stage, all parties know the respective public keys g^{a_i} of (Alice, Bob) and g^{c_i} of (Carole, Daniel). Eventually, all parties output the joint secret key $(g^{a_i})^{c_i} = (g^{c_i})^{a_i}$.

Limitation. The main downside of the protocol of [ACMS23] is that it must reveal a hash of a subset of the bits of the exponent. In turn, this means that security must rely on a strong form of the Diffie-Hellman assumption which provides *leakage* on the exponent, in the form of a hard-to-invert function of (part of) the exponent. Unfortunately, reducing such strong forms of leakage-resilience assumptions to standard assumptions is typically non-trivial, and we are not aware of any standard model choice of H for which such a reduction is known. In fact, *even modeling H as a random oracle*, it does not appear feasible to reduce the assumption underlying the protocol of [ACMS23] to a standard assumption such as the Diffie-Hellman assumption. The work of [ACMS23] sidesteps this limitation by directly proving the security of their protocol in the generic group model, while modeling the hash H as a random oracle. Because Shoup's generic group model implies in particular a random oracle, this yields a 4-NIKE in Shoup's GGM.

Our approach. Our key idea to overcome the limitations of the protocol of [ACMS23] is to remove the leakage altogether from the protocol. Instead, to guarantee a sufficient probability of collisions between Alice and Bob, we *sparsify the exponent*. That is, let $\mathtt{Gen} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a length-doubling injective pseudorandom generator (PRG). Consider the following alternative protocol:

- Alice picks N λ-bit seeds $(s_1, \cdots, s_N)$ for the PRG, and lets $(a_1, \cdots, a_N) \leftarrow (\mathtt{Gen}(s_1), \cdots, \mathtt{Gen}(s_N))$. Alice sends $(g^{a_1}, \cdots, g^{a_N})$ to Bob.
- Bob picks N λ-bit seeds $(t_1, \cdots, t_N)$ for the PRG, and lets $(b_1, \cdots, b_N) \leftarrow (\mathtt{Gen}(t_1), \cdots, \mathtt{Gen}(t_N))$. Bob sends $(g^{b_1}, \cdots, g^{b_N})$ to Bob.

Because the image of $\mathtt{Gen}$ is of size at most 2^λ, the probability of collisions after sending $N = \lambda \cdot 2^{\lambda/2}$ elements is overwhelming. This allows Alice and Bob to immediately agree on a collision. Carole and Deva execute in parallel a similar protocol, and the final key is derived using a Diffie-Hellman instance, as in [ACMS23]. As a bonus, this protocol is conceptually simpler; in particular, Bob and Deva do not need anymore to run the kangaroo algorithm of [Pol75] to compute a discrete logarithm.

Security in the standard model. To prove security, we first introduce the PRG-CDH assumption, to which we reduce the security of the construction. At a high level, the PRG-CDH assumption states that it is infeasible to compute $g^{x \cdot y}$ from (g, g^x, g^y), even when x, y are sampled as the outputs of a pseudorandom generator $\mathtt{Gen}$ (i.e., $x = \mathtt{Gen}(s_x)$ and $y = \mathtt{Gen}(s_y)$ with random seeds s_x, s_y). We separately prove that the PRG-CDH assumption is implied by the security of the PRG and that of CDH, via a straightforward sequence of hybrids.

To reduce security to the PRG-CDH assumption, we embed a PRG-CDH challenge in a simulated execution of the protocol. Here, an important subtlety is that compared to [ACMS23], we must change the way the parties agree non-interactively on a collision (whenever there are more than one collision to choose

from). In [ACMS23], this was done by picking the lexicographically first collision among all colliding pairs. However, this creates complications when trying to embed a PRG-CDH, as the position in the list where we must embed the challenge is not independent of the value of the challenge itself. To circumvent this limitation, we let instead the parties agree to use the first group element from Alice's message $(g^{a_1}, \cdots, g^{a_N})$ which is equal to an element of Bob's message. This makes the selected collision independent of the value of the group element, which considerably simplifies the reduction (and, as a byproduct, allows to formulate the protocol directly in Maurer's model, where the parties are not given a representation of the group elements by bitstrings).

Let $\mathcal{A}$ be an adversary which finds (with non-negligible probability) the shared key K at the end of an execution of the protocol. The reduction proceeds as follows: it receives a PRG-CDH challenge (g, g^x, g^y) and samples messages $(g^{a_1}, \cdots, g^{a_N})$ and $(g^{b_1}, \cdots, g^{b_N})$ for Alice and Bob respectively. Then, it identifies the first g^{a_i} colliding with one of the g^{b_j}, and replaces all occurrences of this group element with g^x. The reduction does the same with Carole and Deva to embed g^y. In this process, it can happen that g^x or g^y collides with other elements (including previous elements from the list of Alice or Carole), in which case the simulation fails (because $\mathcal{A}$ does not solve a CDH problem that involves g^x, g^y). However, it is not too hard to bound the probability of this bad event. Furthermore, we show that conditioned on the bad event not happening, the simulated transcript is perfectly distributed as an honest transcript, hence the reduction solves the PRG-CDH challenge with the same success probability as the adversary.

Security in Maurer's model: challenges. We now turn to the security of our protocol in Maurer's generic group model. Somewhat surprisingly, we observe that our proof in the standard model does *not* imply security in Maurer's GGM. The issue is that in the GGM, adversaries are allowed to run in unbounded time (but are restricted in the number of queries they make to the group oracle). While the reduction to PRG-CDH is generic, the proof of security of PRG-CDH shows that if there exists an adversary $\mathcal{A}$ against PRG-CDH, under the CDH assumption (which holds unconditionally in Maurer's model [MW98]), there is a reduction $R^{\mathcal{A}}$ that turns this adversary into an attacker against the PRG. However, since $\mathcal{A}$ need not to be *efficient* (as a circuit or Turing machine), this does not imply in general, an efficient attack on the PRG! We note that this constitutes, to our knowledge, the first natural example of a primitive whose security in the standard model is easier to prove compared to its security in an idealized model (we note that this is not specific to Maurer's model: the same issue appears with Shoup's model).

Since the generic group model is an idealized model whose purpose is to devise heuristic arguments for security in the real world, one might be tempted to say that considering unbounded attackers is a mere artifact of the original definition, and that the right thing to do would be to simply consider a variant of Maurer's model where the adversaries are also polynomially bounded.

And indeed, our standard model result trivially extended to this "polynomially-bounded Maurer". The issue is that the *negative* result of [ACMS23] does not: it is essential for their impossibility result that the adversary runs in unbounded time, for otherwise, the parties could simply ignore the generic group altogether and run an arbitrary 3-party NIKE (hence, in particular, extending the result of [ACMS23] to this model would rule out 3-party NIKEs altogether). Hence, a significant gap remains between the negative result (in the unbounded Maurer model) and the positive result (in the "polynomially-bounded Maurer model").

Security in Maurer's model: a way around. To close the gap, we show how the security of our protocol extends, non-trivially, to the "standard" Maurer model with unbounded adversaries. Concretely, we show that if there exists a (possibly inefficient) q-query generic adversary $\mathcal{A}$ against PRG-CDH, then there necessarily exists a list L of q triples $(\alpha_i, \beta_i, \gamma_i)_{i \leq q}$ such that with high probability over the choice of two random seeds s_x, s_y, denoting $x \leftarrow \mathtt{Gen}(s_x)$ and $y \leftarrow \mathtt{Gen}(s_y)$, there exists $i \neq j$ such that

$$\alpha_i \cdot x + \beta_i \cdot y + \gamma_i = \alpha_j \cdot x + \beta_j \cdot y + \gamma_j.$$

In contrast, it is relatively straightforward to prove that for *random* x, y, such a collision is unlikely to happen unless q is very large. Then, we use L as a non-uniform advice string, which we hardcode in the circuit of a distinguisher against the PRG. This yields a non-uniform distinguisher of size $O(q)$, contradicting the non-uniform security of the PRG. A limitation of this proof technique, however, is that it only yields a non-uniform distinguisher. Due to two recent breakthrough results [MP23, HIW23] on inverting one-way functions with a non-uniform advice, this implies that our reduction for PRG-CDH only proves its security up to at most $N^{8/5}$ queries (if the bound of [MP23, HIW23] is the best possible).

Discussions. Our result in Maurer's model might appear confusing at first: we prove security of our protocol in a model where the adversary is *unbounded*, yet we rely on the (standard model) security of a pseudorandom generator, which the unbounded adversary can break. There is, however, no contradiction: we prove that if there exists a *generic adversary* that can break the PRG-CDH assumption in Maurer's GGM (instantiated with any sufficiently strong and non-uniformly secure injective PRG), then there exists a *standard model (non-uniform) attacker* that can break the PRG. In particular, this implies that the PRG-CDH assumption can remain secure even against an adversary that can break the underlying PRG (intuitively, this stems from the fact that pseudorandomness is not necessary for the PRG-CDH assumption to hold: in the generic group model, it suffices that the distribution of the exponents, which are random samples from the range of the PRG, has some suitable statistical properties, and the *standard model security* of the PRG suffices to show that it must possess these statistical properties).

We also note that combining pseudorandom generators with Maurer's model might seem conflicting, because in Maurer's model, the parties do not hold any

representation of the group elements they manipulate (they are only stored as entries of a list accessible through oracle queries). We clarify that we still rely on the standard formulation of Maurer's model, and do not extend it or modify it for the purpose of using PRGs: the parties use the PRGs solely to generate pseudorandom *exponents* x (which are, in Maurer's model as in the standard model, simply elements of $\mathbb{Z}_p$), which they load in the list (which corresponds to creating g^x) using the oracle queries allowed by Maurer's model typically, Add queries to run the square-and-multiply algorithm).

2 Preliminaries

Given an integer $N \in \mathbb{N}$, we let $[N]$ denote the set $\{1, \cdots, N\}$. We use bold font to denote vectors. If L is a list of pairs, $L[i][1]$ and $L[i][2]$ denote the first and second elements of the i-th pair. If L is a list and x an element, we denote by $L||x$, and $x||L$ respectively the list L with x added in the rightmost position, and the list L with x added in the leftmost position. If A, B are two sets, we let A^B denote the set of functions from B to A. For an algorithm $\mathcal{A}$, and an input x we let $[\mathcal{A}(x)]$ denote the set $\{y \ : \ \exists r, y = \mathcal{A}(x; r)\}$, where r denote $\mathcal{A}$'s internal randomness. We let $\mathtt{GroupGen}$ denote a a deterministic polynomial-time procedure which takes as input 1^λ and outputs the description of a group $\mathbb{G}(\lambda)$ of prime cardinality $p(\lambda) \in [2^\lambda; 2^{\lambda+1}]$. For a list (or a vector) L, we denote by $\mathrm{Card}(L)$, the number of distinct elements in the list.

2.1 Pseudorandom generators

Below, we define pseudorandom generators. For simplicity, we consider a general definition of PRGs which output elements of a set S (not necessarily of the form $S = \{0,1\}^n$) and whose outputs should appear indistinguishable from random elements of S. Looking ahead, we will rely in this work on PRGs with outputs in $\mathbb{Z}_p^*$ where p is a large prime.

Definition 1 (Pseudorandom generator). *Let $(t, \varepsilon) \in \mathbb{N}^{\mathbb{N}} \times [0;1]^{\mathbb{N}}$, and S a family of finite sets of integers indexed by $\mathbb{N}$. Let $\mathtt{Gen}$ be a PPT algorithm such that for all $\lambda \in \mathbb{N}$ and $s \in \{0,1\}^\lambda$, we get $\mathtt{Gen}(s) \in S(\lambda)$. We say $\mathtt{Gen}$ is a (t, ε, S)-PRG iff for every $t(\lambda)$-time adversary $\mathcal{A}$, and all large enough $\lambda \in \mathbb{N}$,*

$$\left| \mathop{\mathbb{P}}_{x \leftarrow_r \{0,1\}^\lambda} [\mathcal{A}(\mathtt{Gen}(x)) = 1] - \mathop{\mathbb{P}}_{y \leftarrow_r S(\lambda)} [\mathcal{A}(y) = 1] \right| \leq \varepsilon(\lambda).$$

We say that $\mathtt{Gen}$ is an "injective PRG" if $\mathtt{Gen}$ is an injective function and a PRG. Eventually, a "family" of PRGs is a set $\{\mathtt{Gen}_k : \{0,1\}^\lambda \to S(\lambda)\}_{\lambda \in \mathbb{N}, k \in \{0,1\}^\lambda}$. We say that $\{\mathtt{Gen}_k\}_k$ is a (t, ε)-secure family of PRGs if for every $t(\lambda)$-time adversary $\mathcal{A}$, and all large enough $\lambda \in \mathbb{N}$,

$$\left| \mathop{\mathbb{P}}_{k, x \leftarrow_r \{0,1\}^\lambda} [\mathcal{A}(k, \mathtt{Gen}_k(x)) = 1] - \mathop{\mathbb{P}}_{\substack{k \leftarrow_r \{0,1\}^\lambda \\ y \leftarrow_r S(\lambda)}} [\mathcal{A}(k, y) = 1] \right| \leq \varepsilon(\lambda).$$

PRG security with two inputs. For an adversary $\mathcal{A}$, and a PRG algorithm $\mathtt{Gen}$, we define games $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 0)$ and $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 1)$ as in Figure 1, and $\mathsf{Adv}(\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda))$ to be as follows:

$$|\,\mathbb{P}[\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 1) = 1] - \mathbb{P}[\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 0) = 1]|$$

Property 1 *If there exists a (t, ε)-adversary distinguishing the two games $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \cdot](\lambda, 0)$ and $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \cdot](\lambda, 1)$ of Figure 1, then there exists an $(t, \varepsilon/2)$-adversary against the security of the underlying PRG algorithm $\mathtt{Gen}$.*

Proof. The reduction follows with a standard hybrid argument. If there exists an adversary $\mathcal{A}$ that distinguishes between the two games $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 0)$ and $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 1)$ with advantage ϵ in time t, then it should be either of the following two cases:

- With advantage at least $\frac{\varepsilon}{2}$, $\mathcal{A}$ can distinguish between $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 0)$ and the hybrid game $\mathsf{Game}^{\mathsf{dPRGh}}[\mathtt{Gen}, \mathcal{A}](\lambda)$.
- With advantage at least $\frac{\varepsilon}{2}$, $\mathcal{A}$ can distinguish between $\mathsf{Game}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 1)$ and the hybrid game $\mathsf{Game}^{\mathsf{dPRGh}}[\mathtt{Gen}, \mathcal{A}](\lambda)$.

Either way, the existence of an $O(t)$-time adversary against the underlying PRG $\mathtt{Gen}$, with advantage of at least $\frac{\varepsilon}{2}$, immediately follows.

$\underline{\mathsf{Game}}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 0)$:	$\underline{\mathsf{Game}}^{\mathsf{dPRGh}}[\mathtt{Gen}, \mathcal{A}](\lambda)$:	$\underline{\mathsf{Game}}^{\mathsf{dPRG_2}}[\mathtt{Gen}, \mathcal{A}](\lambda, 1)$:
1. $s_0, s_1 \leftarrow_r \{0,1\}^\lambda$	1. $s \leftarrow_r \{0,1\}^\lambda$	1. $x_0, x_1 \leftarrow_r S(\lambda)$
2. $x_0 \leftarrow \mathtt{Gen}(s_0)$	2. $x_0 \leftarrow \mathtt{Gen}(s)$	2. $b \leftarrow \mathcal{A}(x_0, x_1)$
3. $x_1 \leftarrow \mathtt{Gen}(s_1)$	3. $x_1 \leftarrow_r S(\lambda)$	3. **return** b
4. $b \leftarrow \mathcal{A}(x_0, x_1)$	4. $b \leftarrow \mathcal{A}(x_0, x_1)$	
5. **return** b	5. **return** b	

Fig. 1. PRG security experiments. All experiments are implicitly parameterized with the description of the PRG algorithm $\mathtt{Gen}$, and of the adversary $\mathcal{A}$.

2.2 Maurer's Generic group model

We recall below the definition of Maurer's Generic Group Model, taken almost verbatim from [ACMS23].

Definition 2 (Maurer's Generic Group Model (MGGM)). *Let $p \in \mathbb{Z}$ be a positive integer. Let $\mathsf{Arr_P}$ 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.*

2.3 Non-interactive key exchange

Definition 3 (NIKE). *A 4-party non-interactive key-exchange is a protocol between 4 parties where each party* $i \in [4]$ *runs a pair of algorithms* $(\mathsf{Msg}_i, \mathsf{Key}_i)$*, where*

- $\mathsf{Msg}_i(1^\lambda)$ *: Given the security parameter* λ*, the algorithm outputs* $\left(\mathbf{m}^{(i)}, \boldsymbol{\eta}^{(i)}\right)$*, where* $\mathbf{m}^{(i)}$ *is the messages, and* $\boldsymbol{\eta}_i$ *is the (secret) state of party* P_i*.*
- $\mathsf{Key}_i(\boldsymbol{\eta}^{(i)}, (\mathbf{m}^{(j)})_{j\in[4]\setminus\{i\}})$*: Given the secret state* $\boldsymbol{\eta}^{(i)}$ *and the other parties' messages* $\left(\mathbf{m}^{(j)}\right)_{j\neq i}$*, the key algorithm outputs a key* k_i*.*

We say a 4-party key exchange protocol $(\mathsf{Msg}_i, \mathsf{Key}_i)_{i\in[4]}$ *is correct if after the following execution:*

$$\forall i \in [4] : (\mathbf{m}^{(i)}, \boldsymbol{\eta}^{(i)}) \leftarrow \mathsf{Msg}_i(1^\lambda)$$
$$\forall i \in [4] : k_i \leftarrow \mathsf{Key}_i(\boldsymbol{\eta}^{(i)}, (\mathbf{m}^{(j)})_{j\in[4]\setminus\{i\}}).$$

For all $i, j \in [4]$*, we have* $k_i = k_j$*. We say a 4-party key-exchange protocol* $(\mathsf{Msg}_i, \mathsf{Key}_i)_{i\in[4]}$ *is* (n, ε)*-secure against an eavesdropper if for all adversaries* $\mathcal{A}(\lambda)$ *that uses at most* $n = n(\lambda)$ *steps of computations:*

$$\mathbb{P}\left[\mathsf{KeyExchange}[(\mathsf{Msg}, \mathsf{Key}), \mathcal{A}](\lambda) = 1\right] \leq \varepsilon = \varepsilon(\lambda),$$

Where the security game $\mathsf{KeyExchange}[(\mathsf{Msg}, \mathsf{Key}), \mathcal{A}]$ *is represented on Figure 2.*

$\underline{\mathsf{KeyExchange}[(\mathsf{Msg}, \mathsf{Key}), \mathcal{A}](\lambda)}$:

1. **for** $i \in [4]$:
2. $\quad \left(\mathbf{m}^{(i)}, \boldsymbol{\eta}^{(i)}\right) \leftarrow \mathsf{Msg}_i\left(1^\lambda\right)$
3. $k \leftarrow \mathcal{A}\left(\mathbf{m}^{(1)}, \ldots, \mathbf{m}^{(4)}\right)$
4. **return** $\left(k \stackrel{?}{=} \mathsf{Key}_1\left(\boldsymbol{\eta}^{(1)}, \mathbf{m}^{(2)}, \mathbf{m}^{(3)}, \mathbf{m}^{(4)}\right)\right)$

Fig. 2. 4-party non-interactive key-exchange security game in the presence of an eavesdropper.

3 Fine-Grained Non-Interactive Key Exchange in the Standard Model

In this section, we introduce a new 4-party key exchange protocol over a prime-order group $\mathbb{G}$. Compared to the original protocol of [ACMS23], our protocol has the same communication and computation overhead, but is conceptually simpler. Furthermore, while the security of the key exchange protocol of [ACMS23] could only be shown in Shoup's generic group model, we show that our new protocol can be proven secure *in the standard model*. Concretely, we prove the following:

Theorem 4. *Let λ be a security parameter, $\varepsilon \leq 1$ be a constant, and let $\mathbb{G}$ be a group of order $p \geq 2^{2\lambda}$. Define $N \leftarrow \lambda \cdot 2^{\lambda/2}$. Then, under the following assumptions:*

- *For any algorithm $\mathcal{A}$ running in time $N^{2-\varepsilon}$, the advantage of $\mathcal{A}$ against the CDH assumption over $\mathbb{G}$ is $o(1)$, and*
- *There exists a $(N^{2-\varepsilon}, o(1), \mathbb{Z}_p^*)$ injective PRG $\mathtt{Gen} : \{0,1\}^\lambda \to \mathbb{Z}_p^*$,*

there exists a 4-party non-interactive key exchange protocol which is $\big(N^{2-\varepsilon}, o(1)\big)$-secure.

Note that $N^{2-\varepsilon} = \lambda^{2-\varepsilon} \cdot 2^{(1-\varepsilon/2)\lambda} = \mathsf{poly}(\lambda) \cdot 2^{(1-\varepsilon/2)\lambda}$. For example, setting $p \approx 2^{2\lambda}$ in the above theorem yields a protocol secure under the assumption that no algorithm running in time $\ll \sqrt{p}$ can achieve constant advantage against CDH, which is in line with the current state of the art (when instantiating Gen with a suitable elliptic curve). Alternatively, setting $p \geq 2^{\lambda \log \lambda}$ (or even $p \geq 2^{\lambda^{1+\delta}}$ for some small $\delta > 0$), the assumption becomes that no adversary with sub-exponential runtime $\approx 2^{\log p / \log\log p}$ (or $2^{(\log p)^{(1/(1+\delta))}} \approx 2^{(\log p)^{1-\delta}}$) can achieve constant advantage against CDH. This yields a 4-NIKE under a weaker assumption on CDH, at the cost of a $\log \lambda$ increase in communication, and the assumption of an exponentially-secure injective PRG with a longer stretch (from λ to $\lambda \log \lambda$ bits, or to $\lambda^{1+\delta}$).

Remark 5 (Uniform versus non-uniform security). Our construction requires a PRG with near-optimal security, since it must be secure against attackers running in time $\mathsf{poly}(\lambda) \cdot 2^{(1-\varepsilon/2)\lambda}$, while brute-force takes 2^λ evaluations. In the non-uniform setting, it is known that such PRGs cannot possibly exist, due to known time-space tradeoffs for inverting arbitrary functions [MP23, HIW23]. This can be circumvented either by assuming security in the *uniform* setting, or alternatively by relying on a keyed *family* of (injective) pseudorandom generators. In this case, our 4-NIKE requires a common random string to non-interactively distribute the key of the PRG.

Remark 6 (Vanishing advantage versus negligible advantage). Theorem 4 guarantees that any sub-quadratic adversary has vanishing advantage $o(1)$. A more precise bound on the advantage, of the form $1/N^{O(\varepsilon)}$, can be obtained by making a more precise assumption on the advantage of $N^{2-\varepsilon}$-time adversaries against

the PRG and CDH respectively. However, as a function of N, this advantage is never negligible. In the generic group model, it is not too hard to reduce the advantage of the adversary to a negligible function of N, by running (say) $\log^2 N$ independent instances of the 4-NIKE, and XORing the keys obtained by the parties. However, in the standard model, such strong amplification theorems are not known, and are believed to be unlikely to exist. In the setting of Merkle's 2-party (fine-grained) NIKE in the random oracle model, for example, the conjecture that parallel repetition can reduce the advantage to a negligible quantity was dubbed the "dream XOR lemma" in [BGI08]. The same paper also showed the impossibility of proving the dream XOR lemma in a blackbox way. More recently, the dream XOR lemma was actually proven to be false assuming indistinguishability obfuscation [BIK+22]. We conjecture that a similar impossibility would hold for the goal of amplifying security of our protocol by parallel repetitions, and leave as an open question the task of building a (fine-grained) 4-NIKE with negligible adversarial advantage in the standard model.

3.1 Protocol description

For conciseness of notations and consistency in our protocol, we define $\sigma : [4] \to [4]^2 \times [4]^2$ to be a function that indicates the level of interactions between parties. To be more precise, σ will pair the parties in such a way that running the protocol concludes in a consistent key among all parties. In particular, for each party P_i, $\sigma(i)$ would output the following pairs:

$$1 \mapsto ((1,2),(3,4)) \quad 2 \mapsto ((1,2),(3,4)) \quad 3 \mapsto ((3,4),(1,2)) \quad 4 \mapsto ((3,4),(1,2))$$

Our protocol is represented in Figure 3. The protocol bears high-level similarities with the one of [ACMS23]. The message algorithm Msg outputs N group elements along with their discrete logarithms (i.e. outputs of the PRG algorithm $\mathtt{Gen}$) for each party P_i. Note that the set of seeds inputted to the PRG algorithm has cardinality $o(N^2)$, and the group size is $\tilde{\Omega}(N^4)$. The key algorithm running by each party P_i, given the secrets (i.e. discrete logarithms) and the messages of all other parties P_j (for $j \in [4] \setminus \{i\}$), finds the collision between the messages of both pairs of parties using $\mathtt{Coll}$ subroutine, and computes the key using its own secrets.
Let's view the key algorithm Key_1 for party P_1. The algorithm invokes $\mathtt{Coll}$ on the messages of two pairs $(\mathbf{m}^{(1)}, \mathbf{m}^{(2)})$ and $(\mathbf{m}^{(3)}, \mathbf{m}^{(4)})$, and find the collision indices s, s' (which is gonna be the first collision in the list of messages of P_1 and P_3). Having the secrets $\boldsymbol{\eta}_1$, it then computes and outputs the key $x_s^{(1)} \cdot g_{s'}^{(3)}$ which is the shared key among parties.

Maurer Compatibility versus Standard efficiency ($\mathtt{Coll}$ *subroutine).* The $\mathtt{Coll}$ algorithm of Figure 4 takes as input two lists (or vectors) L_1 and L_2, and outputs the first index of the first list L_1 that collides with one of the elements of the second list L_2. Our protocol of Figure 3 uses this subroutine in order to find the first collision (relative to the first list) among the messages of each pairs

$\underline{\mathsf{Msg}_i(1^\lambda)}$:

1. Let $N := 2^{\frac{\lambda}{2}} \cdot \lambda$
2. **for** $j \in [N]$:
3. $\quad s_j \leftarrow_r \{0,1\}^\lambda$
4. $\quad x_j \leftarrow \mathtt{Gen}(s_j)$
5. **for** $j \in [N]$:
6. $\quad$ Let $g_j \leftarrow x_j \cdot g$
7. **return** $(\mathbf{m} = (g_1, \ldots, g_N),$
   $\boldsymbol{\eta} = (x_1, \ldots, x_N))$

$\underline{\mathsf{Key}_i(\boldsymbol{\eta}, (\mathbf{m}^{(j)})_{j \in [4] \setminus \{i\}})}$:

1. Let $\boldsymbol{\eta} = (x_1, \ldots, x_N)$
2. **for** $j \in [4] \setminus \{i\}$:
3. $\quad$ Parse $\mathbf{m}^{(j)} = (g_1^{(j)}, \ldots, g_N^{(j)})$
4. **for** $j \in [N]$:
5. $\quad g_j^{(i)} \leftarrow x_j \cdot g$
6. $((\ell, f), (\ell', f')) \leftarrow \sigma(i)$
7. $j \leftarrow \mathtt{Coll}\left(\mathbf{m}^{(\ell)}, \mathbf{m}^{(f)}\right)$
8. Set s such that $g_j^{(\ell)} = g_s^{(i)}$
9. $s' \leftarrow \mathtt{Coll}\left(\mathbf{m}^{(\ell')}, \mathbf{m}^{(f')}\right)$
10. **if** $\infty \in \{s, s'\}$: **return** $\perp$
11. **else** : **return** $\left(x_s \cdot g_{s'}^{(\ell')}\right)$

Fig. 3. A 4-party non-interactive key-exchange, parametrized by a group $\mathbb{G}$ of size at least $2^{2\lambda}$ and a PRG $\mathtt{Gen}$.

of parties. The left-hand side protocol of Figure 4 is compatible with Maurer's model (where the group elements are not represented by bitstrings) but requires a large number of equality queries (i.e., $\approx N^2$ equality tests). This is not an issue here: in Maurer's paradigm, all the operations that are not considered group operations (including equality test) are typically considered for free. The right-hand side protocol of Figure 4 is sorting the union of the given lists in order to find the colliding index, which can be more relevant for a more standard complexity analysis.

$\underline{\mathtt{Coll}\,(L_1, L_2)}$:

1. Let $N \leftarrow |L_1|$
2. Let $\mathsf{flag} \leftarrow \infty$
3. **for** k from N down to 1:
4. $\quad$ **for** j from N down to 1:
5. $\quad\quad$ **if** $\mathsf{Equal}(L_1[k], L_2[j])$:
6. $\quad\quad\quad \mathsf{flag} \leftarrow k$
7. **return** flag

$\underline{\mathtt{Coll}\,(L_1, L_2)}$:

1. Let $\mathsf{flag} \leftarrow \infty$
2. Let $N \leftarrow |L_1|$
3. Let $L \leftarrow []$
4. **for** $j = 1$ to N
5. $\quad L \leftarrow L || (L_1[j], j)$
6. $\quad L \leftarrow L || (L_2[j], \infty)$
7. Sort L according to the lexicographic order of the first element of each pair.
8. **for** j from 1 to $2N$:
9. $\quad$ **if** $L[j][1] = L[j+1][1]$:
10. $\quad\quad$ **if** $L[j][2], L[j+1][2] \in \mathbb{N} \times \{\infty\}$:
11. $\quad\quad\quad \mathsf{flag} \leftarrow \min(\mathsf{flag}, L[j][2])$
12. **return** flag

Fig. 4. Two approaches for finding the first colliding index of two lists.

3.2 Correctness

It is easy to see that from the construction, if the game does not abort (i.e., no party returns $\perp$), then parties $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \mathsf{P}_4$ agree on two indices (i_{12}, i_{34}) such that all parties know $(g^{(1)}_{i_{12}}, g^{(3)}_{i_{34}})$, $(\mathsf{P}_1, \mathsf{P}_2)$ know $x_{i_{12}}$ such that $x_{i_{12}} \cdot g = g^{(1)}_{i_{12}}$, and $(\mathsf{P}_3, \mathsf{P}_4)$ know $x_{i_{34}}$ such that $x_{i_{34}} \cdot g = g^{(3)}_{i_{34}}$. From there, correctness follows as in the classical Diffie-Hellman key exchange. It remains to show that the game does not abort with high probability. We focus on the parties $(\mathsf{P}_1, \mathsf{P}_2)$ as the analysis is identical for $(\mathsf{P}_3, \mathsf{P}_4)$. Note that $(\mathsf{P}_1, \mathsf{P}_2)$ output $\perp$ if and only if $\mathbf{m}^{(1)}$ and $\mathbf{m}^{(2)}$ share no common element (i.e., $\mathtt{Coll}(\mathbf{m}^{(1)}, \mathbf{m}^{(2)})$ returns ∞). Now, let $\mathbf{s}^{(i)} = (s^{(i)}_1, \ldots, s^{(i)}_N)$ denote the seeds output during the execution of Msg_i. Below, we denote $\mathbf{m}^{(1)}$ and $\mathbf{m}^{(2)}$ the lists of group elements sent by P_1 and P_2 respectively. We note that $s^{(1)}_i = s^{(2)}_j$ is equivalent to $g^{(1)}_i = g^{(2)}_j$, by injectivity of the $\mathtt{Gen}$. The probability to have a collusion on at least one pair of indices of $L_1 \times L_2$ is lower bounded by:

$$\begin{aligned}
\mathbb{P}[\mathbf{m}^{(1)} \cap \mathbf{m}^{(2)} \neq \emptyset] &= \mathbb{P}[\mathbf{s}^{(1)} \cap \mathbf{s}^{(2)} \neq \emptyset] \\
&\geq \mathbb{P}[\mathbf{s}^{(1)} \cap \mathbf{s}^{(2)} \neq \emptyset \wedge \mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}] \\
&\geq \mathbb{P}[\mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}] \cdot \mathbb{P}[\mathbf{s}^{(1)} \cap \mathbf{s}^{(2)} \neq \emptyset | \mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}]
\end{aligned}$$

Let us bound both terms of this product. First :

$$\begin{aligned}
\mathbb{P}[\mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}] &= 1 - \mathbb{P}[\mathrm{Card}(\mathbf{s}^{(1)}) < 2^{\frac{\lambda}{2}}] \\
&\geq 1 - \mathbb{P}\left[\exists S \subset [2^\lambda], \mathrm{Card}(S) = 2^{\frac{\lambda}{2}}, \forall i \in [N], s^{(1)}_i \in S\right] \\
&\geq 1 - \sum_{S \subset [2^\lambda], \mathrm{Card}(S) = 2^{\frac{\lambda}{2}}} \mathbb{P}\left[\forall i \in [N], s^{(1)}_i \in S\right] \\
&\geq 1 - 2^{\lambda 2^{\frac{\lambda}{2}}} \mathbb{P}\left[\forall i \in [N], s^{(1)}_i \in [2^{\lambda/2}]\right] \\
&= 1 - 2^{\lambda 2^{\frac{\lambda}{2}}} \left(\frac{1}{2^{\lambda/2}}\right)^{2^{\frac{\lambda}{2}}\lambda} = 1 - \left(\frac{2}{2^{\lambda/2}}\right)^{2^{\frac{\lambda}{2}}\lambda} \\
&\geq 1 - \mathsf{negl}(\lambda),
\end{aligned}$$

where the second inequality follows by a union bound, the third uses the fact that $\binom{a}{b} \leq a^b$ and that the probabilities $\mathbb{P}\left[\forall i \in [N], s^{(1)}_i \in S\right]$ are identical for all subsets $S \subset \{0,1\}^\lambda$ of the same cardinality $2^{\lambda/2}$, and the fourth from the fact that each $s^{(1)}_i$ is picked uniformly at random over $\{0,1\}^\lambda$. It remains to bound the second term:

$$\begin{aligned}
\mathbb{P}[\mathbf{s}^{(1)} \cap \mathbf{s}^{(2)} \neq \emptyset | \mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}] &= 1 - \mathbb{P}[\mathbf{s}^{(1)} \cap \mathbf{s}^{(2)} = \emptyset | \mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}] \\
&\geq 1 - \mathbb{P}[\forall i \in [N], s^{(2)}_i \notin \mathbf{s}^{(1)} | \mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}] \\
&\geq 1 - \left(\mathbb{P}_{x \leftarrow_r \{0,1\}^\lambda}[x \notin \mathbf{s}^{(1)} | \mathrm{Card}(\mathbf{s}^{(1)}) \geq 2^{\frac{\lambda}{2}}]\right)^N
\end{aligned}$$

$$\geq 1 - \left(1 - \frac{1}{2^{\lambda/2}}\right)^{N} = 1 - \left(\left(1 - \frac{1}{2^{\lambda/2}}\right)^{2^{\lambda/2}}\right)^{\lambda}$$
$$\geq 1 - \left(\frac{1}{e} + o(1)\right)^{\lambda} = 1 - \mathsf{negl}(\lambda).$$

Then we conclude that the probability to have a collusion between party P_1 and P_2 is negligible. And because the distribution for players P_3 and P_4 is exactly the same, we deduce that the protocol fails with a negligible probability. □

3.3 The PRG-CDH assumption

To analyze the protocol of Figure 3, we first introduce an intermediate cryptographic assumption, the PRG-CDH assumption. At a high level, PRG-CDH states that the CDH assumption holds when the random exponents of the CDH security game are replaced by the outputs of a pseudorandom generator. We first show in Theorem 7 that PRG-CDH reduces to the standard CDH assumption under the security of the PRG, and then reduce the security of our protocol to PRG-CDH in Lemma 8. The formal definition of the PRG-CDH security game is represented on Figure 5.

$\underline{\mathsf{Game}}^{\mathsf{dCDH}}[\mathbb{G}, d, \mathcal{A}](\lambda)$:	$\underline{\mathcal{D}}^{\mathsf{prg}}[\mathtt{Gen}](\lambda)$:	$\underline{\mathcal{D}}^{\mathsf{uni}}[p](\lambda)$:
1. $(x_1, x_2) \leftarrow (d(\lambda), d(\lambda))$	1. $s \leftarrow_r \{0,1\}^{\lambda}$	1. $x \leftarrow_r \mathbb{Z}_p^*$
2. $g' \leftarrow \mathcal{A}(g^{x_1}, g^{x_2})$	2. **return** $(\mathtt{Gen}(s))$	2. **return** (x)
3. **return** $(g' = g^{x_1 \cdot x_2})$		

Fig. 5. The CDH game parameterized by a distribution $d = d(\lambda)$. Setting $d = \mathcal{D}^{\mathsf{uni}}[p](\lambda)$ is the standard CDH security game (denoting as $\mathsf{Game}^{\mathsf{CDH}}[\mathbb{G}, \mathcal{A}](\lambda)$), while setting $d = \mathcal{D}^{\mathsf{prg}}[\mathtt{Gen}](\lambda)$ yields the PRG-CDH security game.

Theorem 7. *Let $\mathcal{A}$ be a t-time adversary against $\mathsf{Game}^{\mathsf{dCDH}}[\mathbb{G}, d, \cdot](\lambda)$ where $d = \mathcal{D}^{\mathsf{prg}}[\mathtt{Gen}](\lambda)$ (as in Figure 5). Then, there exists an $\mathcal{O}(t)$-time adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}(\mathsf{Game}^{\mathsf{dCDH}}[\mathbb{G}, \mathcal{D}^{\mathsf{prg}}[\mathtt{Gen}](\lambda), \mathcal{A}](\lambda))$$
$$\leq \mathsf{Adv}(\mathsf{Game}^{\mathsf{CDH}}[\mathbb{G}, \mathcal{A}](\lambda)) + \mathsf{Adv}(\mathsf{Game}^{\mathsf{dPRG}_2}[\mathtt{Gen}, \mathcal{B}](\lambda)).$$

Proof. Construct $\mathcal{B}$ as follows: on input (x, x'), $\mathcal{B}$ invokes the algorithm $\mathcal{A}$ and computes $g' \leftarrow \mathcal{A}(g, g^x, g^{x'})$. It then returns 1 if $g' = g^{x \cdot x'}$, and 0 otherwise. It is easy to see that the running time of $\mathcal{B}$ is of $\mathcal{O}(t)$. Furthermore,

$$\mathsf{Adv}(\mathsf{Game}^{\mathsf{dPRG}_2}[\mathtt{Gen}, \mathcal{B}](\lambda))$$
$$= |\,\mathbb{P}[\mathsf{Game}^{\mathsf{dPRG}_2}[\mathtt{Gen}, \mathcal{B}](\lambda, 1) = 1] - \mathbb{P}[\mathsf{Game}^{\mathsf{dPRG}_2}[\mathtt{Gen}, \mathcal{B}](\lambda, 0) = 1]|$$

$$= |\mathsf{Adv}(\mathsf{Game}^{\mathsf{dCDH}}[\mathbb{G}, \mathcal{D}^{\mathsf{prg}}[\mathtt{Gen}], \mathcal{A}](\lambda)) - \mathsf{Adv}(\mathsf{Game}^{\mathsf{dCDH}}[\mathbb{G}, \mathcal{D}^{\mathsf{uni}}[p(\lambda)], \mathcal{A}](\lambda))|$$

By definition $\mathsf{Adv}(\mathsf{Game}^{\mathsf{dCDH}}[\mathbb{G}, \mathcal{D}^{\mathsf{uni}}[p(\lambda)], \mathcal{A}](\lambda)) = \mathsf{Adv}(\mathsf{Game}^{\mathsf{CDH}}[\mathbb{G}, \mathcal{A}](\lambda))$, that concludes the proof. □

3.4 Reduction to PRG-CDH

Lemma 8. *Suppose there exists a (T, ε)-adversary against the 4-NIKE construction $(\mathsf{Msg}, \mathsf{Key})$ of Figure 3. Then there exists a $\left(T + \tilde{O}(N), \varepsilon \cdot (1 - N/2^{\lambda-2})\right)$-adversary against the PRG-CDH assumption.*

Proof. Let $\mathcal{A}$ be a (T, ε)-adversary against the construction $(\mathsf{Msg}, \mathsf{Key})$ of Figure 3. Consider the following games:

Game_0. In the first game, the challenger emulates the exact execution of the 4-party NIKE from Figure 3 honestly, and hands the transcript to the adversary $\mathcal{A}$, who then returns a key K. In particular, the challenger does as follows:

1. For $i \in [4]$, the challenger:
   - samples $(s_1^{(i)}, \cdots, s_N^{(i)}) \leftarrow_r (\{0,1\}^\lambda)^N$
   - computes $(x_1^{(i)}, \cdots, x_N^{(i)}) \leftarrow (\mathtt{Gen}(s_1^{(i)}), \cdots, \mathtt{Gen}(s_N^{(i)}))$
   - computes $(g_1^{(i)}, \cdots, g_N^{(i)}) \leftarrow (x_1^{(i)} \cdot g, \cdots, x_N^{(i)} \cdot g)$
2. The challenger sends $(g_1^{(i)}, \cdots, g_N^{(i)})$ to $\mathcal{A}$ for all $i \in [4]$.
3. The adversary $\mathcal{A}$ returns a key K.

Let j_{12} be the first index $j \in [N]$ where $s_j^{(1)} = s_k^{(2)}$ for some $k \in [N]$, and j_{34} be the first index $j \in [N]$ where $s_j^{(3)} = s_{k'}^{(4)}$ for some $k' \in [N]$. The adversary wins the game (denoted as the event Win_0), if $K = x_{j_{12}}^{(1)} \cdot g_{j_{34}}^{(3)}$.

Game_1. In this game, the challenger behaves as follows:

1. For $i \in [4]$, the challenger:
   - samples $(s_1^{(i)}, \cdots, s_N^{(i)}) \leftarrow_r (\{0,1\}^\lambda)^N$
   - define $(s_1^{(\mathrm{init},i)}, \cdots, s_N^{(\mathrm{init},i)}) \leftarrow (s_1^{(i)}, \cdots, s_N^{(i)})$ (it will be useful for the security analysis)
   - computes $(x_1^{(i)}, \cdots, x_N^{(i)}) \leftarrow (\mathtt{Gen}(s_1^{(i)}), \cdots, \mathtt{Gen}(s_N^{(i)}))$
   - computes $(g_1^{(i)}, \cdots, g_N^{(i)}) \leftarrow (x_1^{(i)} \cdot g, \cdots, x_N^{(i)} \cdot g)$
2. The challenger sends $(g_1^{(i)}, \cdots, g_N^{(i)})$ to $\mathcal{A}$ for all $i \in [4]$.
3. The challenger computes $j_{12} \leftarrow \mathtt{Coll}\left(\mathbf{s}^{(1)}, \mathbf{s}^{(2)}\right)$, and $j_{34} \leftarrow \mathtt{Coll}\left(\mathbf{s}^{(3)}, \mathbf{s}^{(4)}\right)$. In particular, it identifies the first index $j_{12} \in [N]$ where $s_{j_{12}}^{(1)} = s_k^{(2)}$ for some $k \in [N]$, and the first index $j_{34} \in [N]$ where $s_{j_{34}}^{(3)} = s_{k'}^{(4)}$ for some $k' \in [N]$. Note that if such indices don't exist, the values are going to set to ∞.

4. The challenger then samples new seeds $(\mathbf{s}^*_{12}, \mathbf{s}^*_{34}) \leftarrow_r (\{0,1\}^\lambda)^2$.
5. Computes $g^*_{12} \leftarrow \mathtt{Gen}(s^*_{12}) \cdot g$ and $g^*_{34} \leftarrow \mathtt{Gen}(s^*_{34}) \cdot g$.
6. For $i = 1, 2$, for every $j \in [N]$ where $s^{(i)}_j = s^{(1)}_{j_{12}}$, it sets $x^{(i)}_j \leftarrow \mathtt{Gen}(s^*_{j_{12}})$ and $g^{(i)}_j \leftarrow g^*_{12}$. For $i = 3, 4$, for every $j \in [N]$ where $s^{(i)}_j = s^{(1)}_{j_{34}}$, sets $x^{(i)}_j \leftarrow \mathtt{Gen}(s^*_{j_{34}})$ and $g^{(i)}_j \leftarrow g^*_{34}$.
7. Eventually, the adversary $\mathcal{A}$ returns a key K.

The adversary wins the game (denoted as the event Win_1), if $K = x^{(1)}_{j_{12}} \cdot g^{(3)}_{j_{34}}$.

Let Flag be the event in which either of the following happens:

- there exists $i \in \{1, 2\}$ and $j \in [N]$ where $s^*_{12} = s^{(i)}_j \neq s^{(1)}_{j_{12}}$, or
- there exists $i \in \{3, 4\}$ and $j \in [N]$ where $s^*_{34} = s^{(i)}_j \neq s^{(3)}_{j_{34}}$.

Claim. $\mathbb{P}[\mathsf{Flag}] \leq N/2^{\lambda-2}$.

Proof. Fix any $i \in [4]$ and $j \in [N]$. Then, $\mathbb{P}_{s^* \leftarrow_r \{0,1\}^\lambda}[s^* = s^{(i)}_j] = 1/2^\lambda$. The claim follows by a straightforward union bound on all values of i and j. □

Claim. $\mathbb{P}[\mathsf{Win}_0] = \mathbb{P}[\mathsf{Win}_1 \mid \overline{\mathsf{Flag}}]$.

Proof. We consider the two games Game_0 and $\mathsf{Game}_1|\overline{\mathsf{Flag}}$ (i.e. Game_1 in which the event Flag does not happen), and find the distribution of the $\mathbf{s}^{(i)} = (s^{(i)}_1, \cdots, s^{(i)}_N)$ for $i \in [4]$ in both games. Note that in both games $\left(\mathbf{s}^{(1)}, \mathbf{s}^{(2)}\right)$ and $\left(\mathbf{s}^{(3)}, \mathbf{s}^{(4)}\right)$ are computed independently.
Now consider the distribution of $\mathbf{s}^{(12)} := \left(\mathbf{s}^{(1)}, \mathbf{s}^{(2)}\right)$ in Game_0 and $\mathsf{Game}_1|\overline{\mathsf{Flag}}$. Let $\mathbf{s}^{(\mathrm{init},12)} := \left(\mathbf{s}^{(\mathrm{init},1)}, \mathbf{s}^{(\mathrm{init},2)}\right)$, where $\mathbf{s}^{(\mathrm{init},i)}$ is as in Game_1. Additionally, note that in Game_0, the distribution of $\mathbf{s}^{(12)}$ is the uniform distribution over $\left(\{0,1\}^\lambda\right)^{2N}$.

We start by introducing the following equivalence relation $\equiv$ in $\left(\{0,1\}^\lambda\right)^{2N}$:

$$(\mathbf{s}^{(1)\prime}, \mathbf{s}^{(2)\prime}) \equiv (\mathbf{s}^{(1)\prime\prime}, \mathbf{s}^{(2)\prime\prime}) \iff$$
$$\exists u \in \{0,1\}^\lambda \setminus \left(\{s^{(12)\prime}_i\}_{i\in[N]} \setminus \{s^{(12)\prime}_t\}\right) : \mathbf{s}^{(12)\prime\prime} = \mathtt{Replace}\left(\mathbf{s}^{(12)\prime}, s^{(12)\prime}_t, u\right),$$

where $t := \mathtt{Coll}\left(\mathbf{s}^{(1)\prime}, \mathbf{s}^{(2)\prime}\right)$ and $\mathtt{Replace}\,(\mathbf{s}, a, b)$ is a function which replaces all the occurrences of a by b in the vector $\mathbf{s}$. We also denote $s^{(12)\prime}_\infty = \perp$.
Informally, $(\mathbf{s}^{(1)\prime}, \mathbf{s}^{(2)\prime}) \equiv (\mathbf{s}^{(1)\prime\prime}, \mathbf{s}^{(2)\prime\prime})$ if we have:

$$(\mathbf{s}^{\mathrm{init},(1)}, \mathbf{s}^{\mathrm{init},(2)}, \mathbf{s}^{(1)}, \mathbf{s}^{(2)}) = (\mathbf{s}^{(1)\prime}, \mathbf{s}^{(2)\prime}, \mathbf{s}^{(1)\prime\prime}, \mathbf{s}^{(2)\prime\prime}). \tag{1}$$

in the same execution of $\mathsf{Game}_1|\overline{\mathsf{Flag}}$.

Let $\mathrm{Eq}\,(\mathbf{s})$ to be the equivalence class of a vector $\mathbf{s} \in \left(\{0,1\}^\lambda\right)^{2N}$, and let $\mathbf{s}^* = \left(\mathbf{s}^{(1)*}, \mathbf{s}^{(2)*}\right) \in \left(\{0,1\}^\lambda\right)^{2N}$. Define

$$\mathtt{NewElements}\,(\mathbf{s}^*) := \{0,1\}^\lambda \setminus \left(\{s^{(*)}_i\}_{i\in[N]} \setminus \{s^{(1)*}_t\}\right).$$

where $t = \mathtt{Coll}\left(\mathbf{s}^{(1)*}, \mathbf{s}^{(2)*}\right)$. Observe that

$$\mathrm{Eq}\left(\mathbf{s}^*\right) = \left\{\mathtt{Replace}\left(\mathbf{s}^*, s_t^{(1)*}, u\right)\right\}_{u \in \mathtt{NewElements}(\mathbf{s}^*)}.$$

In particular, if $t = \mathtt{Coll}\left(\mathbf{s}^{(1)*}, \mathbf{s}^{(2)*}\right) = \infty$, then $\mathrm{Eq}\left(\mathbf{s}^*\right) = \{\mathbf{s}^*\}$.
Therefore, we have:

$$\mathbb{P}[\mathbf{s}^{(12)} = \mathbf{s}^* | \overline{\mathsf{Flag}}] = \sum_{\mathbf{s}^{\mathrm{init},*} \in \mathbb{Z}_p^{2N}} \mathbb{P}[\mathbf{s}^{\mathrm{init}} = \mathbf{s}^{\mathrm{init},*} | \overline{\mathsf{Flag}}]\, \mathbb{P}[\mathbf{s}^{(12)} = \mathbf{s}^* | \overline{\mathsf{Flag}} \wedge \mathbf{s}^{\mathrm{init}} = \mathbf{s}^{\mathrm{init},*}]$$

Noting that $\mathbf{s}^{\mathrm{init}} \notin \mathrm{Eq}\left(\mathbf{s}^{\mathrm{init},*}\right) \implies \mathbb{P}[\mathbf{s}^{(12)} = \mathbf{s}^* | \overline{\mathsf{Flag}} \wedge \mathbf{s}^{\mathrm{init}} = \mathbf{s}^{\mathrm{init},*}] = 0$,

$$\mathbb{P}[\mathbf{s}^{(12)} = \mathbf{s}^* | \overline{\mathsf{Flag}}] = \sum_{\mathbf{s}^{\mathrm{init},*} \in \mathrm{Eq}(\mathbf{s}^*)} \mathbb{P}[\mathbf{s}^{\mathrm{init}} = \mathbf{s}^{\mathrm{init},*} | \overline{\mathsf{Flag}}]\, \mathbb{P}[\mathbf{s}^{(12)} = \mathbf{s}^* | \overline{\mathsf{Flag}} \wedge \mathbf{s}^{\mathrm{init}} = \mathbf{s}^{\mathrm{init},*}].$$

Noting that the distribution of $\mathbf{s}^{\mathrm{init}}$ is uniform and independant of $\overline{\mathsf{Flag}}$,

$$\begin{aligned}\mathbb{P}[\mathbf{s}^{(12)} = \mathbf{s}^* | \overline{\mathsf{Flag}}] &= \sum_{\mathbf{s}^{\mathrm{init},*} \in \mathrm{Eq}(\mathbf{s}^*)} \frac{1}{2^\lambda}\, \mathbb{P}[\mathbf{s}^{(12)} = \mathbf{s}^* | \overline{\mathsf{Flag}} \wedge \mathbf{s}^{\mathrm{init}} = \mathbf{s}^{\mathrm{init},*}] \\ &= \sum_{\mathbf{s}^{\mathrm{init},*} \in \mathrm{Eq}(\mathbf{s}^*)} \frac{1}{2^\lambda} \cdot \frac{1}{|\mathrm{Eq}\left(\mathbf{s}^*\right)|} = \frac{1}{2^\lambda},\end{aligned}$$

where the first equality above holds as $\overline{\mathsf{Flag}} \wedge \mathbf{s}^{\mathrm{init}} = \mathbf{s}^{\mathrm{init},*}$ implies the fact that $\mathbf{s}^{(12)}$ is uniformly sampled from $\mathrm{Eq}\left(\mathbf{s}^*\right)$. It follows that the distribution of $\mathbf{s}^{(12)}$ is identical to its distribution in Game_0 (which is the uniform distribution over $\{0,1\}^\lambda$).
The same analysis applies for the distribution of $\mathbf{s}^{(34)}$ (which was independent). Because the distributions are identical, and Win_0 and Win_1 are identical events, thus, the probability of success in Game_0 and $\mathsf{Game}_1 | \overline{\mathsf{Flag}}$ is the same, and this concludes the proof. □

Claim. $\mathbb{P}[\mathsf{Win}_1] = \mathsf{Adv}(\mathsf{Game}^{\mathsf{dCDH}}[\mathbb{G}, \mathcal{D}^{\mathsf{prg}}[\mathtt{Gen}], \cdot](\lambda))$.

Proof. The proof proceeds via a straightforward reduction. Given an adversary $\mathcal{A}$ in Game_1, the reduction $\mathcal{B}$ emulates the challenger in Game_1 except that it does not compute (g_{12}^*, g_{34}^*) by resampling the seeds. Instead, $\mathcal{B}$ receives (g_{12}^*, g_{34}^*) from its challenger, as the challenge of the PRG-CDH experiment. Eventually, $\mathcal{B}$ will output the key K received from $\mathcal{A}$.
The distribution of (g_{12}^*, g_{34}^*) in Game_1 and in the reduction are identical, and the event Win_1 happens iff $K = \mathsf{DLOG}(g_{12}^*) \cdot g_{34}^*$, which concludes the proof. □

We conclude the proof of lemma 8 with a straightforward application of the Bayes rule:

$$\begin{aligned}\varepsilon = \mathbb{P}[\mathsf{Win}_0] &= \mathbb{P}[\mathsf{Win}_1 \mid \overline{\mathsf{Flag}}] \leq \mathbb{P}[\mathsf{Win}_1] / \mathbb{P}[\overline{\mathsf{Flag}}] \\ &\leq \mathsf{Adv}(\mathsf{Game}^{\mathsf{dCDH}}[\mathbb{G}, \mathcal{D}^{\mathsf{prg}}[\mathtt{Gen}], \mathcal{A}](\lambda))/(1 - N/2^{\lambda-2}).\end{aligned}$$

Note that the reduction requires only $\tilde{O}(N)$ computation (assigning a unit cost to operations on group elements). □

3.5 Extensions

We briefly mention a few extensions and generalizations of our protocol. First, as in [ACMS23], our result extends immediately to a 6-party non-interactive key exchange with quadratic security over a group equipped with a bilinear pairing: the generalization groups the 6 parties in three pairs of parties, and each pair of party relies on the same approach as in Figure 3 to agree on a joint key pair. Then, all three pairs run the 3-party bilinear Diffie-Hellman key exchange protocol of Joux [Jou00]. By the same analysis as for our 4-party NIKE, this yields a 6-party NIKE under the existence of exponentially secure injective PRGs and the CDH assumption in bilinear groups:

Corollary 9. *Let λ be a security parameter, $\varepsilon \leq 1$ be a constant, and let $\mathbb{G}$ be a group of order $p \geq 2^{2\lambda}$ equipped with a bilinear pairing $e : \mathsf{Gen} \times \mathsf{Gen} \to \mathsf{Gen}_T$, where Gen_T denotes the target group. Define $N \leftarrow \lambda \cdot 2^{\lambda/2}$. Then, under the following assumptions:*

- *For any algorithm $\mathcal{A}$ running in time $N^{2-\varepsilon}$, the advantage of $\mathcal{A}$ against the CDH assumption over $\mathbb{G}$ is $o(1)$, and*
- *There exists a $(N^{2-\varepsilon}, o(1), \mathbb{Z}_p^*)$ injective PRG $\mathsf{Gen} : \{0,1\}^\lambda \to \mathbb{Z}_p^*$,*

there exists a 6-party non-interactive key exchange protocol which is $\big(N^{2-\varepsilon}, o(1)\big)$-secure.

In addition, other tradeoffs between security and number of parties are possible. For example, triples of parties can directly agree on a key pair (with overwhelming probability) by decreasing the seed length of the PRG, from λ bits to $3\lambda/4$ bits, which yields a 6-party NIKE secure against $N^{1.5-\varepsilon}$-time adversaries under PRG + CDH (without pairings), or even a 9-party key exchange (with pairings). Eventually, because our work gets rid of the idealized assumptions, we anticipate that our techniques should extend to the setting of multiparty NIKE from other assumptions, such as LWE-based or isogeny-based NIKE.

4 Fine-Grained Non-Interactive Key Exchange in Maurer's GGM

In the previous section, we described a 4-party non-interactive key exchange, and proved that it achieves security against near quadratic adversaries, assuming the existence of exponentially hard PRG, and the near-exponential security of the CDH assumption. In this section, we now turn to our second main result: the security of our construction in the generic group model of Maurer.

4.1 On generic versus standard security

The CDH assumption is known to hold unconditionally in Maurer's generic group model [MW98]. Therefore, one might be tempted to conclude directly from our

previous result that, assuming the existence of exponentially secure PRGs, there is a 4-NIKE with near-quadratic security in Maurer's GGM.

Perhaps surprisingly, this claim does in fact *not* follow from our result of the previous section. Recall that our security analysis proceeds in two steps:

1. Prove that the (near-exponential) PRG-CDH assumption holds assuming (1) that the PRG is exponentially secure, and (2) that CDH is near-exponentially secure.
2. Reduce the security of the 4-NIKE to that of PRG-CDH (with a loss proportional to $N/2^\lambda$).

The second step is independent of whether the security of PRG-CDH holds in Maurer's GGM or in the standard model. However, the first step only works in the standard model. In the GGM, we face the following technicality: assuming that CDH is hard, our analysis turns an efficient adversary against PRG-CDH into an efficient adversary against the PRG. However, a generic adversary is only efficient *in the number of queries*. As an oracle Turing machine (or oracle circuit), it can run in unbounded time, provided that its number of queries is bounded. Hence, the reduction, which internally runs the (possibly inefficient) generic adversary, does not yield an efficient attack against the PRG in general.

Before moving on to the contribution of this section, we note that this observation is quite interesting in itself: while one typically expects security in the standard model to be harder to achieve compared to security in the GGM, our construction yields a *natural* example of the opposite phenomenon, showing that the GGM is not strictly stronger than the standard model. This makes sense – in the GGM, adversaries are restricted to being generic, but can also have unbounded runtime, which makes them incomparable to standard model adversaries – but this had not, to our knowledge, been previously observed for any natural cryptographic primitives.

4.2 A way around and a limitation

The failure of this reduction stems from the fact that a "standard" PRG is, in fact, not secure against a generic adversary, that can always run in unbounded time. Unfortunately, it is impossible to build pseudorandom generators in Maurer's GGM. This stands in stark contrast with Shoup's GGM, where the group elements are represented with random strings: there, the group oracle can be used to construct a random oracle, which implies a PRG. Hence, at this stage, one could be tempted to conclude that there is no way around: the PRG-CDH assumption cannot possibly hold against an adversary that can break the underlying PRG.

Looking ahead, we show in this section that this intuition is also flawed: perhaps surprisingly again, we show that either the PRG-CDH assumption holds in Maurer's GGM, or there exists an *efficient* (standard model) attack against the PRG. The downside is that this efficient attack is non-uniform: at a high level, we circumvent the inefficiency of GGM adversaries by hard-coding the queries

that they produce in the attacker circuit. In turn, this implies that our result only proves that PRG-CDH holds in Maurer's GGM assuming the existence of a PRG with exponential security against *non-uniform* adversaries. As we will see, this has severe consequences: unlike in the standard model, we cannot rely on a family of PRGs (as the PRG must be fixed before we can hardcode the queries produced by the GGM adversary). Unfortunately, two recent breakthrough results [MP23, HIW23] established that every *fixed* one-way function with domain $\{0,1\}^\lambda$ can be inverted by a non-uniform circuit of size at most $2^{4\lambda/5}$. This implies that our PRG can at best be assumed to be secure against $2^{2-\varepsilon}$-time adversaries for $\varepsilon > 0.4$, from which we can only conclude the inexistence of attacks in time $\gg N^{1.6}$. This leaves a significant gap between the best-possible provable gap our protocol can achieve in Maurer's GGM and the lower bound of [ACMS23] (which only rules out all $o(N^2)$ adversaries). Towards the end of this section, we discuss the question of closing this remaining gap. We state below the main theorem of this section.

Theorem 10. *Let λ be a security parameter, $\varepsilon \leq 1$ be a constant, and let $\mathbb{G}$ be a generic group of order $p \geq 2^{2\lambda}$. Define $N \leftarrow \lambda \cdot 2^{\lambda/2}$. If there exists a non-uniformly secure $(\tilde{O}(N^{2-\varepsilon}), \mu, \mathbb{Z}_p^*)$ injective PRG $\mathsf{Gen} : \{0,1\}^\lambda \mapsto \mathbb{Z}_p^*$, then any generic adversary making at most $N^{2-\varepsilon}$ to the $(\mathsf{Equal}, \mathsf{Add})$ oracles has advantage at most $\mu + 1/2^\lambda + N^{4-2\varepsilon}/(p-1)$ against the 4-party non-interactive key-exchange protocol of Figure 3.*

We prove Theorem 10 in the following section.

4.3 Security analysis

We first observe that Lemma 8 was proven via a generic reduction, which carries to the Maurer setting. It remains to prove that (under the existence of an exponentially secure injective PRG) the PRG-CDH assumption is secure in Maurer's GGM against adversaries running in time $N^{2-\varepsilon}$. We do so by constructing, assuming the existence of an $N^{2-\varepsilon}$-query generic adversary with advantage δ against the PRG-CDH game, a circuit $\mathcal{C}$ of size $\tilde{O}(N^{2-\varepsilon})$ with advantage at least $\delta - 1/2^\lambda - N^{4-2\varepsilon}/(p-1)$ against the game $\mathsf{Game}^{\mathsf{dPRG}_2}$. From there, Theorem 10 follows immediately from Property 1.

The starting point of our analysis is Shoup's security analysis of the CDH assumption in the GGM (while it was written for Shoup's GGM, the analysis carries immediately to Maurer's GGM). Fix the order $p \in \mathbb{Z}$, a PRG $\mathsf{Gen} : \{0,1\}^\lambda \to \mathbb{Z}_p$, and a query bound q, and let $\mathcal{A}$ be a q-query adversary with advantage δ against the PRG-CDH assumption in the GGM. In what follows, e denotes always the last nonzero index of Arr. We consider two experiments, Game_0 and Game_1, which we describe below.

Game_0. This is the security game for the PRG-CDH assumption in Maurer's GGM. We consider a reduction $R_0^{\mathcal{A}}$ that is given black-box access to $\mathcal{A}$ and honestly emulates the behavior of the group oracle throughout the game and maintains an array $\mathsf{Arr}_{\mathcal{A}}$, initialized at $\mathsf{Arr}_{\mathcal{A}}[1] \leftarrow 1$.

- At the start of the game, R_0 samples two seeds $(s_a, s_b) \leftarrow_r (\{0,1\}^\lambda)^2$. R_0 writes the value $a \leftarrow \mathtt{Gen}(s_a)$ at $\mathsf{Arr}_{\mathcal{A}}[2]$ and the value $b \leftarrow \mathtt{Gen}(s_b)$ at $\mathsf{Arr}_{\mathcal{A}}[3]$.
- The adversary $\mathcal{A}$ submits polynomially-many Add and Equal queries adaptively, out of which at most q are Add queries. R_0 answers as in Maurer's GGM (Definition 2).
- $\mathcal{A}$ wins (and Game_0 outputs 1) iff $\mathsf{Arr}[e] = a \cdot b$.

In parallel, the reduction R_0 maintains a list L of linear polynomials, as follows: define two formal variables (X_a, X_b), and set $L[1] \leftarrow 1$, $L[2] \leftarrow X_a$, and $L[3] \leftarrow X_b$. Whenever R_0 receives a query $\mathsf{Add}(i_1, i_2, c_1, c_2)$ with $(i_1, i_2) \in [e]$ and $(c_1, c_2) \in \mathbb{Z}_p$ from $\mathcal{A}$, it sets $e \leftarrow e + 1$ and $L[e] \leftarrow c_1 \cdot L[i_1] + c_2 \cdot L[i_2]$ (note that this is a linear polynomial in X_a, X_b).

Game_1. In this experiment, we follow Shoup's strategy and let the reduction $R_1^{\mathcal{A}}$ emulate the answers of the group oracle using the list L directly, as follows:

- At the start of the game, the reduction defines two formal variables (X_a, X_b). R_1 maintains an internal list L and writes X_a at $L[2]$ and X_b at $L[3]$. It sets $e = 3$.
- The adversary $\mathcal{A}$ submits polynomially-many Add and Equal queries adaptively, out of which at most q are Add queries.
  - Whenever R_1 receives a query $\mathsf{Add}(i_1, i_2, c_1, c_2)$ with $(i_1, i_2) \in [e]$ and $(c_1, c_2) \in \mathbb{Z}_p$ from $\mathcal{A}$, it sets $e \leftarrow e + 1$ and $L[e] \leftarrow c_1 \cdot L[i_1] + c_2 \cdot L[i_2]$.
  - Whenever R_1 receives a query $\mathsf{Equal}(i_1, i_2)$ from $\mathcal{A}$ with $i_1, i_2 \in [e]$, it returns 1 if $L[i_1] = L[i_2]$ (as formal polynomials) and 0 else.
- Once $\mathcal{A}$ terminates, the game samples two seeds $(s_a, a_b) \leftarrow_r (\{0,1\}^\lambda)^2$ and set $a \leftarrow \mathtt{Gen}(s_a)$, $b \leftarrow \mathtt{Gen}(s_b)$. Let $P(X_a, X_b) \leftarrow \mathsf{Arr}_{\mathcal{A}}[e]$. The game outputs 1 iff $P(a, b) = a \cdot b$.

We let $R_1^{\mathcal{A}}$ output the list L.

Constructing the adversary against $\mathtt{Gen}$. We let E denote the event that, at the end of the game (either Game_0 or Game_1), there exists $i, j \leq e$ such that, denoting $P_i \leftarrow L[i]$ and $P_j \leftarrow L[j]$, $P_i \neq P_j$ yet $P_i(a, b) = P_j(a, b)$. Let $\bar{E}$ denote the complementary event. We prove a few simple claims.

Claim. $\mathbb{P}[\mathsf{Game}_0 \mid \bar{E}] = \mathbb{P}[\mathsf{Game}_1 \mid \bar{E}]$.

Proof. Observe that an answers of R_0 and R_1 to a query $\mathsf{Equal}(i_1, i_2)$ of $\mathcal{A}$ differs between Game_0 and Game_1 only if it holds that $\mathsf{Arr}_{\mathcal{A}}[i_1] \neq \mathsf{Arr}_{\mathcal{A}}[i_2]$, yet $L[i_1] = L[i_2]$, in which case the event E is raised. Therefore, conditioned on the event E not happening, the answers of R_0 and R_1 are perfectly identical in both Game_0 and Game_1, and the winning condition is also identical in both. □

Claim. $\mathbb{P}[\mathsf{Game}_1] \leq 1/2^\lambda$.

Proof. Observe that in Game_1, the answers of R_1 to the queries of $\mathcal{A}$ are perfectly independent of a, b. Let $u \cdot X_a + v \cdot X_v + w \leftarrow L[e]$. We have

$$\begin{aligned}\mathbb{P}[\mathsf{Game}_1] &= \mathbb{P}\left[\begin{array}{l}(s_a, s_b) \leftarrow_r (\{0,1\}^\lambda)^2, \\ (a,b) \leftarrow (\mathtt{Gen}(s_a), \mathtt{Gen}(s_b))\end{array} \;:\; u \cdot a + v \cdot b + w = a \cdot b\right] \\ &= \mathbb{P}\left[\begin{array}{l}(s_a, s_b) \leftarrow_r (\{0,1\}^\lambda)^2, \\ (a,b) \leftarrow (\mathtt{Gen}(s_a), \mathtt{Gen}(s_b))\end{array} \;:\; a = (vb + w) \cdot b^{-1} - u\right] \leq \frac{1}{2^\lambda},\end{aligned}$$

which follows from the fact that $\mathtt{Gen}$ is injective, hence $\mathbb{P}[a = x] \leq 1/2^\lambda$ for any $x \in \mathbb{Z}_p^*$, with equality when x is in the image of $\mathtt{Gen}$. □

Then, we have:

$$\begin{aligned}\mathbb{P}[\mathsf{Game}_0] &= \mathbb{P}[\mathsf{Game}_0 \wedge E] + \mathbb{P}[\mathsf{Game}_0 \wedge \bar{E}] \leq \mathbb{P}[E] + \mathbb{P}[\mathsf{Game}_0 \mid \bar{E}] \cdot \mathbb{P}[\bar{E}] \\ &= \mathbb{P}[E] + \mathbb{P}[\mathsf{Game}_1 \mid \bar{E}] \cdot \mathbb{P}[\bar{E}] = \mathbb{P}[E] + \mathbb{P}[\mathsf{Game}_1 \wedge \bar{E}] \\ &\leq \mathbb{P}[E] + \mathbb{P}[\mathsf{Game}_1] \leq \mathbb{P}[E] + 1/2^\lambda,\end{aligned}$$

Hence $\mathbb{P}[E] \geq \mathbb{P}[\mathsf{Game}_0] - 1/2^\lambda = \delta - 1/2^\lambda$. Furthermore, by a standard averaging argument, we can assume that $\mathcal{A}$ is deterministic (and so is $R_1^{\mathcal{A}}$). In summary, $R_1^{\mathcal{A}}$ produces a list L of length at most q such that with probability at least $\delta - 1/2^\lambda$ over the sampling of s_a, s_b, the list contains two distinct polynomials P_i, P_j such that $P_i(a,b) = P_j(a,b)$ (with $a \leftarrow \mathtt{Gen}(s_a)$ and $b \leftarrow \mathtt{Gen}(s_b)$).

Now, we run $R_1^{\mathcal{A}}$ and get a list L. Let $\mathcal{C}$ be a circuit with the list L hardcoded in its description. On input a pair $(a,b) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$, the circuit $\mathcal{C}$ checks whether there exists two distinct polynomials P_i, P_j in L such that $P_i(a,b) = P_j(a,b)$. It outputs 1 if there are such polynomials, and 0 otherwise.

Claim. The circuit $\mathcal{C}$ distinguishes between two games $\mathsf{Game}^{\mathsf{dPRG}_2}[\mathtt{Gen}, \cdot](\lambda, 0)$ and $\mathsf{Game}^{\mathsf{dPRG}_2}[\mathtt{Gen}, \cdot](\lambda, 1)$ in time $\tilde{O}(q)$ with advantage at least $\delta - 1/2^\lambda - q^2/(p-1)$.

Proof. First, note that $|L| = O(q)$. Evaluating all polynomials in L on input (a,b) and identifying a collision takes time at most $\tilde{O}(q)$ (e.g. by evaluating all polynomials then sorting the evaluations). By construction, we have $\mathbb{P}[\mathcal{C}(a,b) = 1] \geq \delta - 1/2^\lambda$ when (a,b) are sampled as in $\mathsf{Game}^{\mathsf{dPRG}_2}[\mathtt{Gen}, \mathcal{A}](\lambda, 0)$. Furthermore, when (a,b) are sampled uniformly at random from $\mathbb{Z}_p^*$, we have, by using the Schwartz-Zippel lemma [Zip79, Sch80], for any distinct linear polynomials P_i, P_j

$$\mathbb{P}[P_i(a,b) = P_j(a,b)] = \frac{1}{p-1},$$

hence by a straightforward union bound, when $(a,b) \leftarrow_r \mathbb{Z}_p^* \times \mathbb{Z}_p^*$, $\mathbb{P}[\mathcal{C}(a,b) = 1] \leq q^2/(p-1)$. This concludes the proof. □

4.4 On removing the PRG assumption

In this section, we discuss the possibility of removing the PRG assumption given a suitable combinatorial object. For any list $L = (P_1, \cdots, P_q)$ of distinct linear

polynomials $P_i(X_a, X_b) = \alpha_i X_a + \beta_i X_b + \gamma_i$, we let $C(L)$ denote the *collision set* of L, that is, the set

$$\{(u,v) \in \mathbb{Z}_p^2 \ : \ \exists i \neq j \leq q, P_i(u,v) = P_j(u,v)\}.$$

Let $\mathtt{Map} : \{0,1\}^\lambda \to \mathbb{Z}_p^*$ be an efficiently computable injective mapping with the following complexity measure f_q for $\mathtt{Map}$:

$$f_q(\mathtt{Map}) = \max_{L:|L|=q} \frac{|C(L) \cap \mathtt{Map}(\{0,1\}^\lambda)|}{|\mathtt{Map}(\{0,1\}^\lambda)|}.$$

It follows from our analysis of the previous section that $f_q(\mathtt{Map})$ measures the maximum advantage that any q-query generic adversary can have in finding a list of L queries to the oracle that will cause Game_1 to raise the event E, when the PRG is replaced by the mapping $\mathtt{Map}$. Therefore, to remove the PRG from our construction, it would suffice to construct an explicit mapping $\mathtt{Map}$ with $f_q(\mathtt{Map}) = o(1)$ when $q = N^{2-\varepsilon}$.

The goal of constructing such mapping is closely related to the construction of Sidon sets in additive combinatorics [O'B04], but appears more complex. This question was first studied explicitly by Schnorr in [Sch01b], who showed that a *random* mapping from $\{0,1\}^\lambda$ to $\mathbb{Z}_p^*$ satisfies this property with overwhelming probability. In our context, this translates to an explicit unconditional construction of 4-NIKE with quadratic security in the common random string model, but with a common random string of length $\tilde{\Omega}(N^2)$ (i.e., quadratically larger than the runtime of the honest parties), and assuming efficient RAM access to the CRS for the honest parties.

The task of explicitly constructing such mappings with a small description (avoiding the need for a large CRS as in Schnorr's result) was put forth and studied in [MMN06], where its relation to the problem of constructing Sidon sets was also discussed. Unfortunately, the best construction achieved in [MMN06] falls short of providing quadratic security: cast in our language, Theorem 9 of [MMN06] shows a mapping $\mathtt{Map}$ (efficiently sampled from a set of mappings with a short description) which satisfies $f_q(\mathtt{Map}) = o(1)$ whenever $q \leq N^{6/5-\varepsilon}$, provided that $\log p \geq 12 \log N$. In turns, this implies unconditionally the existence of a 4-NIKE in Maurer's model, with security against $N^{6/5-\varepsilon}$-query adversaries.

References

ACMS23. A. Afshar, G. Couteau, M. Mahmoody, and E. Sadeghi. Fine-grained non-interactive key-exchange: Constructions and lower bounds. In *EUROCRYPT 2023, Part I*, *LNCS* 14004, pages 55–85. Springer, Heidelberg, April 2023.

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.

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.

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.

BIK+22. S. Badrinarayanan, Y. Ishai, D. Khurana, A. Sahai, and D. Wichs. Refuting the dream XOR lemma via ideal obfuscation and resettable MPC. In *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, *LIPIcs* 230, pages 10:1–10:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

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.

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.

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.

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.

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

CLT13. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 476–493. Springer, Heidelberg, August 2013.

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

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

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.

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

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.

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.

HIW23. S. Hirahara, R. Ilango, and R. Williams. Beating brute force for compression problems. *Electron. Colloquium Comput. Complex.*, TR23-171, 2023.

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

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.

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

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

MMN06. I. Mironov, A. Mityagin, and K. Nissim. Hard instances of the constrained discrete logarithm problem. In *International Algorithmic Number Theory Symposium*, pages 582–598. Springer, 2006.

MP23. N. Mazor and R. Pass. The non-uniform perebor conjecture for time-bounded kolmogorov complexity is false. *Cryptology ePrint Archive*, 2023.

MW98. U. M. Maurer and S. Wolf. Lower bounds on generic algorithms in groups. In *EUROCRYPT'98*, *LNCS* 1403, pages 72–84. Springer, Heidelberg, May / June 1998.

O'B04. K. O'Bryant. A complete annotated bibliography of work related to sidon sequences. *The Electronic Journal of Combinatorics [electronic only]*, 11, 2004.

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`.

Sch80. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

Sch01a. C. Schnorr. Small generic hardcore subsets for the discrete logarithm: Short secret dl-keys. *Inf. Process. Lett.*, 79(2):93–98, 2001.

Sch01b. C.-P. Schnorr. Small generic hardcore subsets for the discrete logarithm: Short secret dl-keys. *Information processing letters*, 79(2):93–98, 2001.

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

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.

Zip79. R. Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.