Towards Tight Adaptive Security of Non-Interactive Key Exchange

Julia Hesse [1] Dennis Hofheinz* [2] Lisa Kohl [3] Roman Langrehr* [2]

[1] IBM Research Europe - Zurich
jhs@zurich.ibm.com
[2] ETH Zurich
{hofheinz,roman.langrehr}@inf.ethz.ch
[3] Cryptology Group, CWI Amsterdam
lisa.kohl@cwi.nl

Abstract

We investigate the quality of security reductions for non-interactive key exchange (NIKE) schemes. Unlike for many other cryptographic building blocks (like public-key encryption, signatures, or zero-knowledge proofs), all known NIKE security reductions to date are non-tight, i.e., lose a factor of at least the number of users in the system. In that sense, NIKE forms a particularly elusive target for tight security reductions.

The main technical obstacle in achieving tightly secure NIKE schemes are adaptive corruptions. Hence, in this work, we explore security notions and schemes that lie between selective security and fully adaptive security. Concretely:

We exhibit a tradeoff between key size and reduction loss. We show that a tighter reduction can be bought by larger public and secret NIKE keys. Concretely, we present a simple NIKE scheme with a reduction loss of $O(N^2 \log(\nu)/\nu^2)$, and public and secret keys of $O(\nu)$ group elements, where N denotes the overall number of users in the system, and ν is a freely adjustable scheme parameter.

Our scheme achieves full adaptive security even against multiple "test queries" (i.e., adversarial challenges), but requires keys of size $O(N)$ to achieve (almost) tight security under the matrix Diffie-Hellman assumption. Still, already this simple scheme circumvents existing lower bounds.

We show that this tradeoff is inherent. We contrast the security of our simple scheme with a lower bound for all NIKE schemes in which shared keys can be expressed as an "inner product in the exponent". This result covers the original Diffie-Hellman NIKE scheme, as well as a large class of its variants, and in particular our simple scheme. Our lower bound gives a tradeoff between the "dimension" of any such scheme (which directly corresponds to key sizes in existing schemes), and the reduction quality. For $\nu = O(N)$, this shows our simple scheme and reduction optimal (up to a logarithmic factor).

We exhibit a tradeoff between security and key size for tight reductions. We show that it is possible to circumvent the inherent tradeoff above by relaxing the desired security notion. Concretely, we consider the natural notion of semi-adaptive security, where the adversary has to commit to a single test query after seeing all public keys. As a feasibility result, we bring forward the first scheme that enjoys compact public keys *and* tight semi-adaptive security under the conjunction of the matrix Diffie-Hellman and learning with errors assumptions.

We believe that our results shed a new light on the role of adaptivity in NIKE security, and also illustrate the special role of NIKE when it comes to tight security reductions.

Keywords: Tight reductions, non-interactive key exchange, pairings, learning with errors.

*Supported in part by ERC CoG grant 724307.

1 Introduction

Non-interactive key exchange (NIKE). A non-interactive key exchange (NIKE) scheme assigns any two users P_i, P_j in a system a common shared key $K_{i,j}$. This assignment should happen without any communication, and be based only on a setup like a public-key infrastructure. A well-known example of a NIKE is the original Diffie-Hellman key exchange scheme [16], in which any party has a public key g^{x_i} with associated secret key x_i, and the shared key for parties with public keys g^{x_i}, g^{x_j} is computed as $K_{i,j} = g^{x_i x_j}$. For security, we would like that $K_{i,j}$ remains hidden to an outsider, i.e., without knowing any of the two involved secret keys.

NIKE schemes have been studied as an explicit cryptographic building block by Cash, Kiltz, and Shoup [11], followed by a more in-depth study of NIKE security notions and corresponding schemes by Freire, Hofheinz, Kiltz, and Paterson [22]. There are a variety of different NIKE schemes from various computational assumptions (e.g., [16, 11, 22, 5, 43, 28, 27]), and a number of NIKE applications including wireless networks [10], deniable authentication [17], and interactive key exchange [6].[1]

NIKE and tight security. One interesting particularity of NIKE schemes is the fact that it seems difficult to *tightly* reduce their security to a standard computational assumption. All known security reductions for NIKE schemes (against adaptive corruptions and to non-interactive assumptions in the standard model) lose a factor of at least N, the overall number of users in the system.[2] In fact, two works by Bader, Jager, Li, and Schäge [2] and Hesse, Hofheinz, and Kohl [28] give lower bounds (of $\mathcal{O}(N^2)$, resp. $\mathcal{O}(N)$) on the reduction loss of large classes of NIKE schemes and reductions.

This is quite remarkable, since for most other cryptographic building blocks (such as public-key encryption and digital signatures [29], zero-knowledge proofs [26], or interactive key exchange [1]), tight security proofs are known even in a multi-user setting. But apart from being a theoretical curiosity, this also means that currently, NIKE keysizes should be chosen rather conservatively, in order to account for a potential security loss in scenarios with a large number of users.

Meta-reductions, and what makes tight NIKE security particularly hard to achieve. The mentioned works [2, 28] already give an indication of what the main technical obstacle to a tight NIKE reduction is. Namely, they employ a meta-reduction [4] that turns any reduction that is "too successful" (i.e., suffers only from a low reduction loss) into a stand-alone problem solver. We give more details on this technique in our technical overview below. This meta-reduction technique has been applied also to other settings (like digital signatures [12], key encapsulation [2], and hierarchical identity-based encryption [34]), but it always hinges on one crucial requirement on the investigated scheme and reduction.

To explain this crucial requirement, assume for concreteness a given NIKE security reduction Λ that is "too successful". A meta-reduction requires that Λ is of a special form, namely that Λ essentially simulates the whole NIKE security experiment (including corruptions) for any NIKE adversary that is given in a black-box way. Furthermore, in this simulation, Λ must be "committed" early on to the secret state of this simulation, and in particular to all NIKE shared keys, even if these shared keys are not revealed during the simulation. The reason for this "committed" requirement will become clearer below, but intuitively it enables a "rewinding attack" on the reduction Λ itself.[3]

Now NIKE and other cryptographic primitives differ in this technical requirement for the applicability of meta-reductions. Namely, for primitives like public-key encryption (PKE), it is relatively easy to construct a reduction that is not committed to its secret state in the above sense. To see why this is the case, observe that in a PKE setting, different user secret keys or ciphertexts are not correlated: corrupting one user (or decrypting one ciphertext) gives no information about other users' secret keys (or the decryption of other ciphertexts). Hence, a reduction that answers corruption or decryption queries does not commit itself to, e.g., decryption of a challenge ciphertext in any way.

On the other hand, corrupting one party P_i in a NIKE scheme immediately reveals all shared keys $K_{i,j}$ that P_i has with other (yet-uncorrupted) parties P_j. This also determines the secret keys of such P_j

[1] In this work, we focus on the public-key setting, i.e., we assume a public-key infrastructure. We note, however, that NIKE has also been considered in the identity-based setting [47, 20, 39].

[2] This means that we can currently only map NIKE adversaries with success probability ε and runtime t to adversaries on a suitable computational assumption with runtime $t' \approx t$ but success probability no more than $\varepsilon' \approx \varepsilon/N$.

[3] In a nutshell, the meta-reduction extracts enough shared keys from Λ to take the role of a successful adversary in a rewound Λ-instance. If Λ is "too successful", this causes Λ to solve the underlying computational problem with these extracted keys. Hence, Λ solves the underlying problem essentially by interacting with itself.

to the extent that the $K_{i,j}$ computed with these not-yet-revealed keys are fixed. Hence, corrupting parties will gradually determine the secret state of a simulation in a NIKE reduction (i.e., the functionality of secret keys of yet-uncorrupted parties). This problem does not appear in, say, PKE or signature schemes, and circumventing this "committing" property in NIKE schemes currently seems to be out of reach of known techniques.

This work: beyond linear security loss. Motivated by this difficulty, in this work we examine this "committing" property closer for a general class of group-based NIKE schemes and slightly relaxed security notions. Specifically, for N again denoting the overall number of parties in the system we ask:

> *For which security notions can we obtain NIKE schemes with a security reduction to a standard assumption with a sublinear loss of $o(N)$?*

We obtain positive and negative results:

- We start off with a simple and intuitive "inner-product-based" NIKE scheme $\mathtt{NIKE}_{\mathsf{ip}}$ that enjoys full adaptive security and offers an interesting tradeoff between security loss and key sizes. Specifically, $\mathtt{NIKE}_{\mathsf{ip}}$ is parameterized by $\nu > 0$, has public and secret keys that comprise $\mathcal{O}(\nu)$ group elements, and a reduction loss of $\mathcal{O}(N^2 \log(\nu)/\nu^2)$ to the matrix Diffie-Hellman assumption [21] (a relaxation of the decision linear assumption) in pairing-friendly groups. In particular, it is possible to set $\nu = N$ to obtain a scheme with an (almost) tight reduction to a standard computational assumption, but which also suffers from large keys.

  While the scheme itself is not very efficient for large ν, it shows a conceptually simple way to conduct a "non-committing" reduction. Essentially, our reduction does not have the problematic "committing" property discussed above because each secret key contains enough entropy to be not quite determined by up to ν corruptions of arbitrary other users. This means that previous lower bounds [2, 28] do not apply to this scheme.

  We also note that our $\mathtt{NIKE}_{\mathsf{ip}}$ is the first to obtain tight security against multiple (i.e., up to ν) "test queries", i.e., adversarial challenges. This essentially means that the scheme guarantees the security of not only a single, but many shared keys even after a number of adaptive corruptions. While this property is implied with polynomial loss by security with respect to a single test query, previous reductions (including the previously "most tightly" secure scheme from [28]) did not consider multiple test queries.

  One can view our result also as a feasibility result about the possibility of tight *bounded* security (much like the notion of bounded chosen-ciphertext security for PKE schemes [14]) for NIKE schemes.

- Next, we demonstrate that this tradeoff between reduction loss and key sizes is to some extent inherent when trying to achieve adaptive NIKE security. Concretely, we show that a large class of group-based NIKE schemes (that includes the original Diffie-Hellman scheme, as well as variations such as the scheme from [28] and our $\mathtt{NIKE}_{\mathsf{ip}}$) must become "committed enough" after ν corruptions whenever keys are of size $o(\nu)$ group elements.[4]

  Our result manifests the tradeoff between key sizes and reduction loss of $\mathtt{NIKE}_{\mathsf{ip}}$, and in fact for a large and natural class of NIKE schemes. We stress that the previous lower NIKE bounds [2, 28] do not offer similar tradeoffs, since they did not consider key sizes at all.

- Finally, motivated by the previous tradeoff, we investigate ways to achieve tight security with (asymptotically) compact keys by relaxing the desired security notion. We find a different tradeoff, and now trade security for tightness. Namely, we construct a NIKE $\mathtt{NIKE}_{\mathsf{sa}}$ with keys whose size do not depend on N, and with an (almost) tight security reduction that however only achieves semi-adaptive security. By "semi-adaptive security", we mean that an adversary is restricted not in the type or number of corruptions, but in the timing and number of test queries (i.e., challenges). Concretely, an adversary may not ask any test query after a certain, a-priori bounded number of ν corruptions (or queries for shared keys between honest parties) have been made. Semi-adaptive

[4] Our formal statement does not involve key sizes directly, but instead the dimension of a certain vector space. However, for all existing schemes there is a direct correspondence between the dimension of that vector space and key sizes, and in fact it seems unclear how to break that correspondence. We give more details below.

security interpolates between a mild form of selective security (in which an adversary has to commit in advance to the parties whose shared keys it wants to be challenged on) and full adaptive security.

Our semi-adaptively secure $\mathtt{NIKE}_{\mathsf{sa}}$ uses $\mathtt{NIKE}_{\mathsf{ip}}$ above as a conceptually simple building block, and additionally relies on FHE techniques. Its security can be reduced (with logarithmic loss) to the conjunction of the matrix Diffie-Hellman problem and the learning with errors (LWE) problem [45].

We believe that this result shows that even if we cannot achieve full adaptive security with compact keys and tightly, we are not limited to merely selective security.

1.1 Technical overview

Setting. Formally, a NIKE is a tuple of algorithms ($\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey}$), where $\mathtt{Setup}$ generates public parameters, $\mathtt{KeyGen}$ on input of the public parameters returns a key pair (pk, sk), and $\mathtt{SharedKey}$ on input of the public parameters, a public key pk_i and a secret key sk_j returns a shared key $K_{i,j}$. Correctness requires that for all honestly generated key pairs we have $K_{i,j} = K_{j,i}$.

Security model. The simplest NIKE security notion to achieve is *selective security*, where the adversary commits to the key pair of users to be challenged (i.e. for which the adversary either receives the real shared key or a random key) *before* seeing any public key. To model realistic attack scenarios, what we would like to capture in the security notion is *fully adaptive security* (also called *CKS-heavy security* [22] after the inventors Cash, Kiltz and Shoup of the notion [11]). Here, the adversary can arbitrarily query oracles $\mathcal{O}_{\mathsf{extr}}$, $\mathcal{O}_{\mathsf{revH}}$ and $\mathcal{O}_{\mathsf{test}}$. $\mathcal{O}_{\mathsf{extr}}$ models the adversary's ability to corrupt a user and reveals the corresponding secret key and $\mathcal{O}_{\mathsf{revH}}$ models the ability of the adversary to observe shared keys in the system and reveals the shared keys between two users. Finally, the purpose of $\mathcal{O}_{\mathsf{test}}$ is to model that an adversary should still not be able to distinguish the (non-revealed) shared keys between any pair of uncorrupted users from random. More precisely, $\mathcal{O}_{\mathsf{test}}$ given a tuple of users either returns the real shared key between the users or a random key (depending on an initially flipped bit). Giving the adversary the power to ask corruption queries adaptively poses a challenge for the security reduction. Consider for example the Diffie-Hellman key exchange. There, public key/ secret key tuples are of the form (g^x, x) and a shared key is computed as $(g^{x_i})^{x_j} = (g^{x_j})^{x_i}$. Thus, the reduction either *knows* x – and therefore cannot make use of an adversary distinguishing shared keys involving x from random – or *does not know* x, and can therefore not answer with the secret key if the adversary decides to corrupt the user.

From selective to adaptive security with loss $\Omega(N^2)$. This can be solved by partitioning proofs, reducing the adaptive security to selective security. More precisely, the reduction guesses the "test query" of the adversary (i.e., the parties involved in the query that the adversary tries to distinguish from random) ahead of time and embeds the underlying challenge only in the two corresponding public keys. The problem of this approach is the security loss: With N overall users in the system, this strategy will only be successful with probability $1/N^2$. This means that the security guarantee decreases when the number of users in the system grows, which one has to account for by choosing larger concrete parameters (e.g. group sizes). Further, an upper bound on the number of users might not be known at the time of setup. In this paper we therefore aim for directly proving adaptive security.

Relaxing the security notion: Semi-adaptive security. We introduce the notion of *ν-semi-adaptive*, which lies in between selective and adaptive security: Here, the adversary has to ask *all* test queries within the first ν-corruptions (but can ask arbitrary extract and reveal queries later), where any user involved in a extract, reveal or test-query counts as one corruption. In the special case of 2-semi-adaptive security the adversary has to commit to a single test query after seeing all public keys.

Security with dishonest key registration (DKR). The security experiments described so far do not give the adversary the opportunity to register keys dishonestly, i.e., publish arbitrary public keys that are not necessarily in the image of $\mathtt{KeyGen}$. This can of course occur in realistic scenarios and is ultimately the security notion to aim for. In this paper we restrict ourselves to security with honest key registration as described above, since the difficulty of constructing NIKEs with tight security occurs when going from selective to adaptive security, rather than going from HKR to DKR security. In fact, using standard methods one can *tightly* transform an HKR-secure NIKE into a DKR-secure one, basically by adding a simulation-sound proof of knowledge of the secret key to the public key (see e.g. [11, 28]).

	$\|pk\|$	Sec. model	$\mathcal{O}$(Sec. loss)	Assumption	Pairing
Diffie–Hellman [16]	$1 \times \mathbb{G}$	HKR	N^2	DDH	-
HPS-based [28]	$3 \times \mathbb{G}$	1-test-HKR	N	DDH	-
CKS08 [11]	$2 \times \mathbb{G}$	DKR	N^2	CDH (ROM)	-
FHKP13 [22]	$1 \times \mathbb{Z}_n$	DKR	N^2	Fact. (ROM)	-
FHKP13 [22]	$2 \times \mathbb{G} + 1 \times \mathbb{Z}_p$	DKR	N^2	DBDH	asymm.
HPS-based [28]	$12 \times \mathbb{G}$	1-test-DKR	N	DLIN	symm.
ν-dim $\mathtt{NIKE}_{\mathsf{ip}}$ (Sec. 3)	$(\nu + 2) \times \mathbb{G}$	HKR	$(N/\nu)^2 \log \nu$	DLIN	symm.
N-dim $\mathtt{NIKE}_{\mathsf{ip}}$ (Sec. 3)	$(N + 2) \times \mathbb{G}$	HKR	$\log N$	DLIN	symm.
ν-dim $\mathtt{NIKE}_{\mathsf{sa}}$ (Sec. 5)	$\nu \cdot$ poly	ν-semi-ad.	$\log N$	DLIN, LWE	symm.
2-dim $\mathtt{NIKE}_{\mathsf{sa}}$ (Sec. 5)	poly	semi-ad.	$\log N$	DLIN, LWE	symm.

Table 1: Comparison of existing NIKE schemes. $|pk|$ denotes the size of the public keys, measured in numbers of group elements and exponents. HKR and DKR denote fully adaptive security [22] with honest and dishonest key registrations (where 1-test-HKR/1-test-DKR refers to the corresponding notion in the single-test-query setting). N denotes the number of parties the adversary interacts with, $2 \leq \nu \leq N$ is arbitrary and poly is a polynomial independent of ν and N. Further, note that losses of the constructions from [11] and [22] stem from applying a generic transformation to level the security guarantees of compared schemes. DDH and CDH correspond to the decisional and computational Diffie-Hellman assumption, ROM stands for random oracle model and "Fact." for Factoring. DBDH stands for decisional bilinear Diffie-Hellman, DLIN for Decision Linear and LWE for Learning With Errors. Finally, note that in all cases DLIN can be replaced by the 2-Matrix Decision Diffie-Hellman assumption (MDDH). More generally, we can build on the k-MDDH assumption at the cost of increasing the public key size and security loss by a factor of k.

	Diffie-Hellman KE	HPS-based KE [28]	**Ip-based $\mathtt{NIKE}_{\mathsf{ip}}$ (Sec. 3)**
BJLS [2]	$\Omega(N^2)$	-	-
HHK [28]	$\Omega(N)$	$\Omega(N)$	-
This work (Sec. 4)	$\Omega(N)$	$\Omega(N)$	$\Omega(N/\nu)$

Table 2: Lower bounds on the security loss of NIKE. Here, the public keys of $\mathtt{NIKE}_{\mathsf{ip}}$ are of size $O(\nu)$. Our lower bound only applies to the HPS-based NIKE [28] when instantiated with the decisional Diffie-Hellman-based hash proof system [15]. We note that (in settings where it applies) the lower bound of [28] gives better constants than ours. We highlight the best known lower bound for each construction in green.

Related work. We give a comparison of our result with previous work in Tables 1 and 2. In order to explain the challenges when constructing tightly secure NIKE, in the following we give a brief explanation of previous techniques used to give upper and lower bounds on tightly secure NIKE.

We first recall the *commitment problem* that occurs when proving security of the Diffie-Hellman NIKE. Namely, the reduction either *knows* a secret key or *does not know* a secret key, since each group element has a unique discrete logarithm. Building on the ideas put forward by Coron [13], Bader, Jager, Li, and Schäge [2] presented a lower bound on the tightness of NIKE schemes for which public keys are *fully* committing to their secret keys and therefore their shared keys. Generally, the idea of a meta-reduction is to turn a "too successful" reduction into a stand-alone problem solver for the underlying (non-interactive) cryptographic assumption. The meta-reduction of Bader, Jager, Li, and Schäge [2] systematically rewinds the reduction Λ to run with all N^2 possible pairs of challenge users, arguing that in any run the reduction *either has to abort* or *indeed return the unique secret key*. Now, if the reduction does not abort with probability larger than $1/N^2$ (i.e., the reduction does not abort on at least 2 out of the $\binom{N}{2}$ possible runs), it follows that one can extract *all* secret keys from the reduction, and thereby perfectly simulate an external "perfect" adversary. (Note that for this to be true it is crucial that the reduction is limited to giving out *unique* secret keys, and therefore the shared keys are also unique.) Altogether, this shows that whenever the reduction is successful with probability larger than $1/N^2$, it could have solved the underlying problem itself. Since this is a contradiction to the hardness of the underlying assumption, it shows that the security loss of $\Omega(N^2)$ for Diffie-Hellman (and, more generally, NIKEs with "committing" public keys) is inherent.

Bypassing the commitment problem with semi-functional public keys. Hesse et al. [28] showed how to bypass the lower bound by allowing to switch to non-committing public keys. Essentially, their scheme allows to introduce "semi-functional" public keys which are computationally indistinguishable from public keys produced by KeyGen. This allows a reduction to escape the fully committed setting by introducing semi-functional public keys that do not necessarily fix the shared key with other (semi-functional or normal) public keys in the system. Their construction still suffers from a security loss of $\Omega(N)$, since their semi-functional public keys do not have secret keys and can thus be recognized upon corruption. Since a reduction needs to plant at least one such public key in order to escape full shared key commitment, a security loss of N is inherent. This lower bound on the security loss was formally shown in [28] for all schemes where normal public keys are committing *and* can be efficiently recognized given a corresponding secret key.

Considering weaker NIKE security notions. By allowing an arbitrary number of adaptive test queries but no corruptions, as was done e.g. in [11], tight security turns out easy to achieve. In fact, even the standard Diffie-Hellman key exchange can be shown (almost) tightly secure with respect to this notion, by simply embedding the underlying challenge into all public keys. Tight security (with a loss of factor $O(\log N)$) then follows by the re-randomizability of the decisional Diffie-Hellman assumption. However, going from test-query-only to adaptive security with corruption introduces a security loss of $\Omega(N^2)$. Since we are not aware of a tighter reduction for the scheme of [11] in the setting of adaptive security with corruptions, we do not consider their scheme tight in the sense of our paper.

Hesse et al. [28] consider a restriction of the above described security notion where the adversary is only allowed a single test query (but at any point of time). Since the generic reduction from the single-test-query setting to the multi-test-query introduces an overhead of $\Omega(N^2)$, in this paper we focus on the multi-test-query setting.

Technical idea 1: Overcoming binding public keys

Our construction. In this work we overcome the limitation of [28] with a NIKE scheme $\mathtt{NIKE}_{\mathsf{ip}}$ where both normal and semi-functional public keys have corresponding secret keys. Our construction is based on symmetric pairing groups. Let g be a group generator of the source group. We write $[x]$ for g^x and for a matrix $\mathbf{M} = (m)_{i,j}$ we write $[\mathbf{M}]$ for $([m])_{i,j}$. The public parameters of our NIKE are

$$\mathsf{pp} := ([\mathbf{D}], [\mathbf{MD}]),$$

where $\mathbf{D}$ is a uniformly random $(\nu+2)\times 2$ matrix and $\mathbf{M}$ is a uniformly random symmetric $(\nu+2)\times(\nu+2)$ matrix. The parameter $\nu \in \mathbb{N}_{\geq 2}$ will become important in the security proof. An normal key pair is now generated as follows: We sample a uniformly random 2-dimensional vector $\mathbf{w}$ and set

$$\mathsf{pk} := [\mathbf{Dw}] \qquad \text{and} \qquad \mathsf{sk} := [\mathbf{MDw}].$$

The shared key between two users is the inner product of one user's public key and the other user's secret key, computed with the pairing. To see that correctness holds, let $(\mathsf{pk}_1 = [\mathbf{Dw}_1], \mathsf{sk}_1 = [\mathbf{MDw}_1])$ and $(\mathsf{pk}_2 = [\mathbf{Dw}_2], \mathsf{sk}_1 = [\mathbf{MDw}_2])$ be two honestly generated key pairs. Then

$$\begin{aligned}\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_1, \mathsf{sk}_2) &= e([\mathbf{w}_1^\top \mathbf{D}^\top], [\mathbf{MDw}_2]) = [\mathbf{w}_1^\top \mathbf{D}^\top \mathbf{MDw}_2]_T = [\mathbf{w}_2^\top \mathbf{D}^\top \mathbf{M}^\top \mathbf{Dw}_1]_T \\ &\overset{(*)}{=} [\mathbf{w}_2^\top \mathbf{D}^\top \mathbf{MDw}_1]_T = e([\mathbf{w}_2^\top \mathbf{D}^\top], [\mathbf{MDw}_1]) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_2, \mathsf{sk}_1).\end{aligned}$$

The equality $(*)$ uses the symmetry of $\mathbf{M}$.

One can interpret the public parameters by setting $(\mathbf{d}_1|\mathbf{d}_2) := \mathbf{D}$ as two exemplary key pairs

$$\mathsf{pp} := ((\mathsf{pk}_1 = [\mathbf{d}_1], \mathsf{sk}_1 = [\mathbf{Md}_1]), (\mathsf{pk}_2 = [\mathbf{d}_2], \mathsf{sk}_2 = [\mathbf{Md}_2])).$$

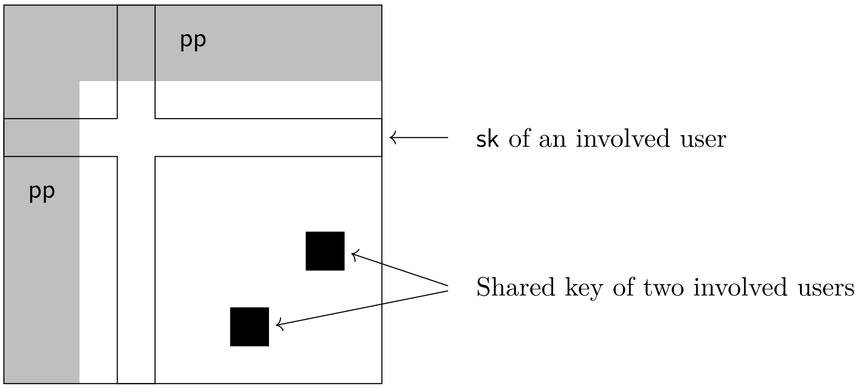

Figure 1: The symmetric matrix $\mathbf{M}$ in the basis where the column vectors of $\mathbf{D}$ and the semi-functional public keys of the involved users are the standard basis vectors. The normal secret keys (and shared keys where at least one user has a normal public key) live in the gray area. The pp can be seen as two key pairs and normal public and secret keys are linear combinations of these public and secret keys.

The user-generated keys are then random linear combinations of these exemplary key pairs. It is necessary to have at least two exemplary keys, because if the honest user keys would be linear combinations of just one exemplary key, one could use the pairing to check efficiently if a public key is in the subspace spanned by the exemplary public key. This would make it impossible for our reduction to use public keys that are not in the linear span of the exemplary public keys.

Semi-functional public keys with secret keys. To argue security, we have to introduce semi-functional public and secret keys. A semi-functional public key is $[\mathbf{u}]$ where $\mathbf{u}$ is chosen uniformly at random from the full space (instead of only the linear span of $\mathbf{D}$'s column vectors). Accordingly, the corresponding semi-functional secret key is $[\mathbf{Mu}]$.

The semi-functional key pairs are indistinguishable from the normal key pairs by the matrix decisional Diffie-Hellman (MDDH) assumption. It states that vectors (represented in a group) from a 2-dimensional subspace (i.e. our normal keys) are indistinguishable from uniformly random vectors (i.e. our semi-functional keys). The MDDH assumption is implied by the well-known 2-linear assumption [21]. Due to the random self-reducibility of the MDDH assumptions, this implication holds even for arbitrary many vectors with security loss only $\mathcal{O}(\log \nu)$.

The semi-functional keys have the desired "less committing" property. Indeed, note that with publishing the public parameters the reduction is not completely committed to the matrix $\mathbf{M}$, since $\mathbf{MD}$ contains only little information about $\mathbf{M}$. Now for each semi-functional public key $\mathbf{u}$ the corresponding semi-functional secret key $\mathbf{Mu}$ leaks some new information about $\mathbf{M}$ and after ν secret keys have been used, the reduction is completely committed to $\mathbf{M}$. If we would apply a suitable basis change transformation to $\mathbf{M}$ (such that the column vectors of $\mathbf{D}$, and the used semi-functional secret keys become unit vectors), each semi-functional secret key corresponds to a row (due to the symmetry also a column) of $\mathbf{M}$ and each shared key corresponds to one entry of the matrix, as depicted by Figure 1.

Since in our scheme there are secret keys for the semi-functional public keys, it circumvents the main bottleneck of the approach of [28]: Our reduction turns all public keys into semi-functional ones, and does not have to rely on any guessing argument. In contrast to [28], our semi-functional keys are committing with respect to normal keys. But, since we turn all keys to semi-functional, it is completely sufficient that semi-functional keys are not committing with respect to other semi-functional public keys. This approach is summarized in Table 3.

Limiting the number of involved users. When ν semi-functional secret keys have been leaked, (i.e., they have been leaked through an $\mathcal{O}_{\mathsf{extr}}$ query or used to answer an $\mathcal{O}_{\mathsf{revH}}$ or $\mathcal{O}_{\mathsf{extr}}$ query,) the reduction is completely committed to $\mathbf{M}$. In this situation we can still argue, that each test query leaked one entry of the matrix $\mathbf{M}$ that was not revealed in any other query and therefore looks uniformly random to the adversary. However, any further leakage of another semi-functional secret key could potentially leak one of the test-query entries. Thus we have to limit the adversary to involve at most ν users in the security

	pk_j normal	pk_j semi-functional
pk_i normal	committed committed	not committed committed
pk_i semi-functional	not committed committed	does not exist Up to ν users involved: not committed Beyond: committed

Table 3: Effect of all combinations of normal and semi-functional public keys on the shared key $K_{i,j}$ in the HPS-based NIKE [28] and our $\mathtt{NIKE}_{\mathsf{ip}}$.

game, where a user counts as involved, when he appeared in at least one $\mathcal{O}_{\mathsf{extr}}$, $\mathcal{O}_{\mathsf{revH}}$, or $\mathcal{O}_{\mathsf{test}}$ query. (Users that have only been registered, i.e., only their public key was revealed, do not count as involved.)

We call the security notion that works like the adaptive security, but where the adversary is allowed to involve at most ν users, ν-bounded security. Even though this security notion is not very realistic, it is a helpful tool because it captures the level of adaptivity that $\mathtt{NIKE}_{\mathsf{ip}}$ can achieve and it implies full adaptive security with security loss only $\mathcal{O}((N/\nu)^2)$. Thus, in total $\mathtt{NIKE}_{\mathsf{ip}}$ can be proven adaptively secure with loss $\mathcal{O}((N/\nu)^2 \log \nu)$. This gives us a tradeoff between key size and tightness. The smaller we select the parameter ν, the smaller the size of the matrix $\mathbf{M}$. This gives us smaller keys, but the semi-functional keys will become committing earlier in the security game, leading to a larger security loss.

A curiosity of $\mathtt{NIKE}_{\mathsf{ip}}$ is that the roles of the public key and secret key are completely symmetric. That is, when all users swap their public and secret key, $\mathtt{NIKE}_{\mathsf{ip}}$ is still secure (and in the security proof we simply have to replace $\mathbf{M}$ by $\mathbf{M}^{-1}$).

Our scheme bypasses the lower bound of [28], because their lower bound requires, informally speaking, that whenever two key pairs look like valid to the adversary, the shared key between them is already determined by the public keys. This is not the case here: Two secret keys could differ by an entry of $\mathbf{M}$ that is unknown to adversary (thus both look like corresponding secret keys for the same public key), but, with a suitable public key of another valid key pair, this entry of $\mathbf{M}$ does not cancel out in the secret key computation and thus the two secret keys yield different shared keys.

Technical idea 2: Lower bound for large class of NIKEs

Inner-product NIKE and a new argument for committing reductions. To extend the existing results on lower bounds, we need to further broaden the class of NIKE schemes that the meta-reduction technique works for. The goal is to allow potential reductions to introduce keys that are less "committing" than in the previous bounds described above. Towards this goal, we observe that all DH-like NIKE schemes in the literature, including our $\mathtt{NIKE}_{\mathsf{ip}}$ described above, have the following joint property: public and secret keys can be represented as $\mathbb{Z}_q^d$-vectors $\mathbf{x}, \mathbf{y}$, and shared keys are computed as (an invertible function of) the inner product $\langle \mathbf{x}, \mathbf{y}' \rangle$. We call such NIKE schemes *d-dimensional ip-NIKE.* The Diffie-Hellman key exchange, for example, allows for key pair (g^x, x) to be written as tuple (x, x) of the same one-dimensional vector $x \in \mathbb{Z}_q$. Shared keys between vector tuples $(x_i, x_i), (x_j, x_j)$ are computed as $(g^{x_i})^{x_j} = g^{\langle x_i, x_j \rangle} = g^{\langle x_j, x_i \rangle} = (g^{x_j})^{x_i}$. Intuitively, using only one-dimensional vectors as in DH-KE means that public keys commit already to all shared keys. Vectors of higher dimensions, though, allow a reduction to encode more information, and eventually escape a setting where all shared keys are fixed. We can now formalize this intuition by exploiting linearity of the inner product. Namely, for a d-dimensional inner-product NIKE, a meta-reduction can create a fully committed setting in case vector dimensions are smaller than the number of users. For this, assume unique[5] public key vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ of $m \approx N$ corrupted users and public key vectors $\mathbf{x}, \mathbf{x}'$ for yet uncorrupted $\mathsf{pk}, \mathsf{pk}'$. Let further $\mathbf{y}_1, \ldots, \mathbf{y}_m, \mathbf{y}, \mathbf{y}'$ denote corresponding secret key vectors. We stress that the meta-reduction is not able to compute any of these values, and we only use them to argue that the reduction is committed. If d is smaller than m, $\mathbf{x}$ lies in the span of the m other vectors with noticeable probability, yielding $\sum_{i=1}^{m} \beta_i \mathbf{x}_i = \mathbf{x}$ for a

[5] For our results we require uniqueness of a corresponding *public* key vector given the public key, which holds for all DH-based schemes from the literature including our first NIKE.

$\mathbb{Z}_q^m$-vector β. This already determines the (exponent of the) shared key $\langle \mathbf{x}, \mathbf{y}' \rangle$ between $\mathbf{x}$ and $\mathbf{x}'$ as a linear combination of the (exponents of) shared keys between each $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and $\mathbf{x}'$. To see this, we write

$$\langle \mathbf{x}, \mathbf{y}' \rangle = \langle \sum_{i=1}^{m} \beta_i \mathbf{x}_i, \mathbf{y}' \rangle = \sum_{i=1}^{m} \beta_i \langle \mathbf{x}_i, \mathbf{y}' \rangle = \sum_{i=1}^{m} \beta_i \langle \mathbf{y}_i, \mathbf{x}' \rangle,$$

where the latter equality follows from the correctness of the NIKE. Since the reduction already committed to the m shared key exponents $\langle \mathbf{y}_i, \mathbf{x}' \rangle_{i \in [m]}$ through corruptions of $\mathsf{pk}_1, \ldots, \mathsf{pk}_m$, we can conclude that the secret key between pk and pk' is fixed through its exponent $\langle \mathbf{x}, \mathbf{y}' \rangle$. We refer the reader to the "uniqueness lemma" (Lemma 4.5) for full details.

A meta-reduction can exploit this committed setting by rewinding the reduction, a technique that was already used to prove the previous lower bounds [2, 28]. And indeed, we can show that any tight reduction must have key dimensions close to N, in order to avoid the linear dependencies described above that would result in commitment of all shared keys in the span. We now describe our meta-reduction and resulting lower bound in detail.

Our new lower bound. We are now ready to explain our lower bound. The general strategy of a meta-reduction is to first describe an inefficient "hypothetical" adversary $\mathcal{A}$ with success probability $\varepsilon_{\mathcal{A}}$, and then show that the hypothetical adversary can be efficiently simulated by rewinding the reduction except when some event "bad" occurs. Since the reduction has to work with the hypothetical adversary, this means that – except with probability $\Pr[\mathsf{bad}]$ – the reduction must also work with the simulated adversary, i.e., without external help. Since by assumption the reduction on its own cannot have more than negligible advantage in solving the underlying problem, this essentially shows that the success probability of the reduction can be upper bounded by $\Pr[\mathsf{bad}] \cdot \varepsilon_{\mathcal{A}} + \mathrm{negl}$ for a negligible function negl, i.e. lose a factor of $1/\Pr[\mathsf{bad}]$. For arguing that the simulated adversary perfectly simulates the hypothetical adversary we crucially rely on the uniqueness lemma, which ensures that all shared keys are fixed after the reduction gave out sufficiently many secret keys.

2-*step-adaptive security.* For proving our lower bound we introduce the 2-*step-adaptive* security notion, where an adversary after receiving the public keys can first ask the secret keys for an arbitrary large set D, and then has to commit to a challenge tuple of public keys (outside D). The adversary wins if after receiving the remaining secret keys (except the ones involved in the challenge tuple), it returns the shared key between the challenge parties. It is straightforward to see that adaptive security implies this weaker security notion, and therefore any lower bound on 2-step-adaptive security readily carries over to adaptive security.

The hypothetical adversary. The idea of the hypothetical adversary is to enforce uniqueness of the challenge shared key by choosing the set D in a suitable way. By the uniqueness lemma this can be achieved by choosing D such that the corresponding public key vectors span all public keys. (Note that if a `NIKE` is a d-dimensional inner-product NIKE, there always exist such a set of size at most d.) Once the shared key between the challenge key pairs is fixed, the adversary can simply brute-force any tuple of secret keys corresponding to the challenge public keys that are consistent with all secret keys in D, and use these to compute the shared key. Since the shared key is unique, the hypothetical adversary will always be successful.[6]

Simulating the hypothetical adversary. The problem in simulating the hypothetical adversary is that the following cannot be done efficiently:

(1) *Extract* the public key vectors to find a spanning subset D, and

(2) Obtain the secret keys by *brute-force* to compute the challenge shared key.

The strategy of the meta-reduction to is therefore to:

(1) *Guess* a set D (and hope it is spanning), and

(2) Obtain secret keys by *rewinding the reduction* to compute the challenge shared key.

[6] To capture adversaries with arbitrary success probability $\varepsilon_{\mathcal{A}}$, the hypothetical adversary can simply flip a biased coin and only output the shared key with probability $\varepsilon_{\mathcal{A}}$.

It turns out productive to choose $|D| \approx N/2$. The reason for this is as follows: On the one hand, for maximizing the probability that D is spanning, D should be chosen as large as possible. On the other hand, for extracting the secret keys from the reduction it is crucial that the reduction can be rewound while already being committed to the secret keys in D (since otherwise, the reduction could give out secret keys that are not consistent with the secret keys in D). In order to argue that the reduction either has to return valid secret keys for each $i \in [N]\backslash D$ or abort with high probability, we have to choose $[N]\backslash D$ large (essentially, the success probability will scale with $1 - 1/(N - |D|)$).

Success probability of the simulated adversary. Finally, the meta-reduction can compute the shared key with the help of this extracted secret keys. By the uniqueness lemma we obtain that this shared key is unique if both strategies of the meta-reduction are successful, i.e. if (1) D is indeed *spanning*, and (2) the reduction returns *valid and consistent secret keys* for both public keys involved in the challenge.[7] We can show that the event bad that either of these is not satisfied only occurs with probability in the order of d/N. This results in the following informal theorem:

Theorem (Lower bound): Any simple reduction from a non-interactive complexity assumption to the adaptive-security of a d-dimensional inner-product NIKE has to lose a factor in the order of $\Omega(N/d)$.

Our lower bound thus yields that $\mathtt{NIKE}_{\mathsf{ip}}$, which is a ν-dimensional ip-NIKE (see Definition 4.4 for the formal definition and Section 4.2 for why $\mathtt{NIKE}_{\mathsf{ip}}$ meets it), with secret keys of size $O(\nu)$ has an inherent security loss of at least $\Omega(N/\nu)$. We contrast that with the security loss of our security proof for the core NIKE, which is $O((N/\nu)^2 \log \nu)$. Thus, for $\nu = N$ the security reduction that we give in Section 3 is essentially optimal. We give a comparison of our lower bound with others in Table 2.

Technical idea 3: Extension to "semi-adaptive" security

Motivation: controlling entropy leakage. The lower bound just presented appears to limit what we can prove about our first NIKE scheme $\mathtt{NIKE}_{\mathsf{ip}}$. Specifically, it appears that we require a large setting of ν (i.e., large keys) for (almost) a tight security reduction. Taking a step back, the intuitive reason why we cannot obtain a better reduction is the following: every secret key revealed through a corruption query leaks entropy about the hidden matrix $\mathbf{M}$. This is intended, since in fact this fresh entropy is used to statistically blind shared keys. However, since the entropy contained in $\mathbf{M}$ is limited, this argument guarantees fresh entropy only for a bounded number of corruptions. After $\mathcal{O}(\nu)$ corruptions, $\mathbf{M}$ is fully determined, and any *additional* corruptions (or shared key or test queries) will result in (jointly) non-uniform shared keys. In particular, the security argument breaks down completely if more than $\mathcal{O}(\nu)$ corruptions are made, even if those are made after all shared key or test queries.

Our goal: ν-semi-adaptive security. We now set out to mitigate this limitation, and better *control* the entropy released through secret keys. We will unfortunately not be able to achieve full adaptive security with small keys. Instead, our goal will be a NIKE scheme with small keys, but in which more than $\mathcal{O}(\nu)$ corruptions are possible only *after* all test queries have been made. To be more concrete: we will achieve what we call ν-semi-adaptive security, which denotes security against the following type of attacks. An adversary may request up to ν corruptions, shared key, or test queries (in any combination). After that, any number of corruption or shared key queries, *but no test queries* are allowed. This notion is hence weaker than adaptive security, but also does allow for some degree of ("early") adaptivity. Like our basic scheme $\mathtt{NIKE}_{\mathsf{ip}}$, our ν-semi-adaptively secure scheme $\mathtt{NIKE}_{\mathsf{sa}}$ will have keys of size $\mathcal{O}(\nu)$ group elements, and its security reduction will be (almost) tight, i.e., only lose a factor of $\mathcal{O}(\log \nu)$.

As discussed above, our result can also be seen as a tradeoff between security and key size: the larger its keys are, the closer to (full) adaptive security the achieved security notion is. We reach full adaptive security only with large keys (of size $\mathcal{O}(N)$ group elements), but smaller keys still yield a less adaptively but (almost) tightly secure scheme.

Building block: non-interactive tag exchange. We now explain the main technical ideas of our semi-adaptively secure $\mathtt{NIKE}_{\mathsf{sa}}$. In a nutshell, we use $\mathtt{NIKE}_{\mathsf{ip}}$ as a *tag generator*, or as what we call a "non-interactive tag exchange" (NITE) scheme. A NITE is defined like a NIKE, except that (a) we

[7] Even though only one secret key is necessary to compute the shared key, we can only be sure that the reduction is committed to the shared key when given both secret keys, since the reduction could switch to a semi-functional public key (without valid secret key).

call shared keys "tags" now, and (b) we require "ν-programmability" instead of indistinguishability for security. ν-programmability requires that there is a dedicated "programming algorithm" that allows to semi-adaptively program tags in the following way: given up to ν pairs of parties $(P_{i,1}, P_{i,2})$ and corresponding "target tags" T_i, output corresponding secret keys that yield T_i as tag between $P_{i,1}$ and $P_{i,2}$. This programming succeeds even *after* all public keys are fixed, and in an adaptive way (such that the T_i can be fixed one at a time, depending on all public keys and earlier T_i). For security, we require that this programming is not detectable, even given *all* secret keys (programmed or not). We can interpret $\mathtt{NIKE}_{\mathsf{ip}}$ as a NITE: shared keys are interpreted as tags, and programming works by adjusting $\mathbf{A}$ adaptively so that the desired tag values are computed.[8] Note that this process works only for programming up to $\mathcal{O}(\nu)$ tag values, since the entropy in $\mathbf{A}$ is limited. On the other hand, the notion of programmability also captures the security that $\mathtt{NIKE}_{\mathsf{ip}}$ achieves when eventually all secret keys are revealed.

Leveraging NITE programmability. The security of a NITE scheme requires programmable tags, but does not require "unopened" tags to remain hidden in any way (e.g., in the sense of NIKE indistinguishability). Hence, we cannot immediately use a NITE scheme as NIKE. Instead, our $\mathtt{NIKE}_{\mathsf{sa}}$ uses a NITE scheme to generate common (but not necessarily secret) shared tags for any two parties, who will then employ a "tag-based NIKE" (TNIKE) as a second stage to compute the actual NIKE shared keys. Analogously to tag-based encryption [32], a TNIKE is simply a NIKE in which shared key computation takes a tag as additional input. For correctness of $\mathtt{NIKE}_{\mathsf{sa}}$ in the usual sense, this tag should of course be the same for both parties.

Before describing a concrete TNIKE scheme, we describe its crucial abstract property: our TNIKE scheme has "punctured" secret keys, i.e., secret keys that allow to compute shared keys for all but one tag value. This puncturing point (i.e., the tag upon which shared key computation fails) is uniformly random, but not obvious from the corresponding public key. Similar puncturing techniques have been used as a technical tool to achieve adaptive security in various contexts before (e.g., [19, 38, 7, 42, 48, 31, 46]). In our security proof, we will program the tags output by the NITE scheme such that *all tags that refer to NIKE test queries will be programmed to be exactly the puncturing points of the corresponding secret keys*.[9] This programming is not detectable thanks to the NITE's security, and leads to a situation in which all test queries are randomized.

Our concrete construction. Armed with this intuition, we now give more details on our actual TNIKE construction. To illustrate the main ideas, we only describe a slightly simplified version of our construction for minimal ν, i.e., such that it achieves only a small degree of semi-adaptivity. The construction is based on the learning with errors (LWE) problem, and assumes public parameters $\mathsf{pp} := \mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_p^{n \times m}$. A public key contains

$$\mathsf{pk} := (\mathbf{SA} + \mathbf{E}, \mathbf{V} = \mathbf{AU} + \tau\mathbf{G}),$$

where $\mathbf{S}$ is a random matrix, $\mathbf{E}$ and $\mathbf{U}$ are a "noise" matrices with small entries, $\mathbf{G}$ is the fixed "gadget matrix" of [36], and τ is the (uniformly random) tag at which the corresponding secret key will be punctured. Note that $\mathbf{V}$ is actually an encryption of τ under the fully homomorphic encryption (FHE) scheme of Gentry, Sahai, and Waters [24].[10] The corresponding secret key is of the form

$$\mathsf{sk} := (\mathbf{S}, \mathbf{U}, \tau).$$

To compute the shared key between two users, assume a public key pk as above, a secret key $\mathsf{sk}' = (\mathbf{S}', \mathbf{U}', \tau')$ from another user, and a tag T. We first homomorphically and deterministically compute an FHE encryption $\mathbf{V}^\star = \mathbf{AU}^\star + b\mathbf{G}$ from $\mathbf{V}$, where $b \in \{0, 1\}$ with $b = 1$ iff $\tau = T$. (Note that this really denotes the punctured point τ encrypted in $\mathbf{V}$, not the one from sk'. Hence, b is hidden at this point.) The corresponding shared key K is a rounded version of $\mathbf{S}'\mathbf{V}^\star$, i.e.,

$$K = \mathsf{round}(\mathbf{S}'\mathbf{V}^\star).$$

[8] This is a slight oversimplification. In fact, programming requires to also make public keys semi-functional, as in the security proof of $\mathtt{NIKE}_{\mathsf{ip}}$ sketched above. Our formal programmability definition will allow for such adjustments during programming.

[9] This is again an oversimplification: for a particular choice of tag, one involved party P_i will not be able to compute the TNIKE shared key, while the other party P_j will be able to compute a shared key that depends on entropy in P_j's secret key.

[10] In this overview, we neglect the fact that τ should be a small scalar. Our full scheme will actually encrypt τ bitwise.

The other involved party, using $\mathsf{pk}' = (\mathbf{S}'\mathbf{A} + \mathbf{E}', \mathbf{V}')$ and $\mathsf{sk} = (\mathbf{S}, \mathbf{U}, \tau)$, computes the same shared key differently: it uses $\mathbf{U}$ to obtain the encryption random coins $\mathbf{U}^\star$ with $\mathbf{V}^\star = \mathbf{A}\mathbf{U}^\star + b\mathbf{G}$ (for b as above) and computes

$$K' = \mathsf{round}((\mathbf{S}'\mathbf{A} + \mathbf{E}')\mathbf{U}^\star) = \mathsf{round}(\mathbf{S}'\mathbf{A}\mathbf{U}^\star + \mathbf{E}'\mathbf{U}^\star) \overset{(*)}{=} \mathsf{round}(\mathbf{S}'\mathbf{A}\mathbf{U}^\star),$$

where $(*)$ holds with high probability for a suitable rounding function, since $\mathbf{E}'$ and $\mathbf{U}$ have small entries. Indeed, $K = K'$ whenever $T \neq \tau$ (so that $b = 0$ and $\mathbf{V}^\star = \mathbf{A}\mathbf{U}^\star$). But for $T = \tau$, the rounded value

$$\mathbf{S}'\mathbf{V}^\star = \mathbf{S}'\mathbf{U}^\star + \mathbf{S}'\mathbf{G}$$

in K contains the term $\mathbf{S}'\mathbf{G}$, which extracts randomness from $\mathbf{S}'$ (that, using a proper setup of $\mathbf{A}$, does not appear in pk'). Hence, the tag $T = \tau$ is special, in that K is randomized by entropy from $\mathbf{S}'$ only for this T. Note that the value K' does not contain this extra term, and so in fact does not satisfy $K' = K$ for $T = \tau$. Of course, since in "normal operation", tags are independently and uniformly random values, $T = \tau$ happens only with negligible probability, and this affects correctness of the scheme only negligibly.

Before going further, we note that this overview over our TNIKE scheme neglects a few things: we did not discuss suitable dimensions, the rounding function, or a suitable encoding of large tags τ. Besides, we did not discuss a generalization to larger values of ν (which require programming more values τ_i into each key). Finally, we did not discuss how both parties coordinate on their role in the computation of K (i.e., on whose $\mathbf{V}$ is used as a basis of computation). All of those questions have simple, albeit sometimes tedious technical answers, and we will discuss all of these issues inside.

The security of our construction. We now briefly sketch the proof of 1-semi-adaptive security of $\mathtt{NIKE}_{\mathsf{sa}}$, which is composed of our NITE and TNIKE schemes. So assume a 1-semi-adaptive adversary $\mathcal{A}$ that obtains all public keys, and then may ask a single test query. After this, and without loss of generality, $\mathcal{A}$ obtains all secret keys of parties not involved in that test query. We need to show that $\mathcal{A}$'s success in distinguishing between real and random answers is negligible, and, for a tight reduction, does not scale in the number of users. To do so, consider the following short sequence of game hops:

Game 0 is the original 1-semi-adaptive NIKE security game with a test query that is answered with the real shared key.

Game 1 changes how the tags for the test query are computed: here, the tag for the test query is adaptively programmed to be the puncturing point τ of the corresponding user. Note that the corresponding shared key can still be computed for $\mathcal{A}$ in the same way that K is computed above. By programmability of the NITE scheme, and using the security of the used FHE scheme, this change goes unnoticed by $\mathcal{A}$.[11]

Game 2 replaces the result of the test query with an independently chosen random shared key. This change is statistical, and can be justified with the observations above about hidden entropy in $\mathbf{S}'$.

Observations about controlling entropy leakage. Taking a step back, we can finally observe how the leakage of entropy is controlled in our composed scheme. In our TNIKE scheme, secret keys carry additional entropy (in the form of $\mathbf{S}$) that is not contained in the corresponding public key. However, accessing and using this additional entropy is somewhat problematic, since we have to preserve correctness and coordinate the two ways to compute the same shared key. In our scheme, setting the tag to the punctured point of a given public key makes that public key act as a randomness extractor: in combination with other secret keys, this public key now extracts randomness from these secret keys. Note that we concede correctness once we use such a punctured point as tag: the secret key whose punctured tag we use computes invalid keys K'.

Hence, the programmability of the NITE scheme gives us the flexibility to (in a bounded way) control which shared keys should be randomized by the additional entropy from one of those parties. Depending on the adaptivity features of the NITE, this can be done even semi-adaptively or fully adaptively. Of course, our own NITE only enjoys ν-programmability (which leads to ν-semi-adaptive security). It is an interesting open question, however, to what extent more flexible NITE schemes are possible.

[11] We note that to obtain *tight* security at this point, we will temporarily switch the used FHE scheme into a lossy mode of encryption [3, 25].

Another interesting question is whether it is possible to implement our strategy with weaker tools (and in particular without relying on FHE techniques). Our concrete technique of secret keys which are punctured at hidden "puncturing points" uses an equality test inside an FHE encryption. It is not clear at all how to implement this kind of puncturing with, say, only inner products instead of FHE.

2 Preliminaries

We use $x \xleftarrow{\$} S$ to denote the process of sampling an element x from a set S uniformly at random. For a probability distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that the random variable x is distributed according to $\mathcal{D}$. If $\mathcal{A}$ is a (probability) algorithm then we write $x \xleftarrow{\$} \mathcal{A}(b)$ to denote the random variable x outputted by $\mathcal{A}$ on input b. We use $\mathsf{Sym}_n(\mathbb{Z}_q)$ (for $n \in \mathbb{N}$, q prime) to denote the set of symmetric $n \times n$ over $\mathbb{Z}_q$. Initially, all partial maps (denoted by $f : A \dashrightarrow B$) are totally undefined in our games. We write $x[i]$ for the i-th bit of the binary representation of x. We write $(a, \_) := (x, y)$ and $(\_, b) := (x, y)$ to define $a := x$ and $b := y$, respectively. $T(\mathcal{A})$ denotes the running time of $\mathcal{A}$.

2.1 Pairing group assumptions

Throughout this paper, $\mathtt{SymGGen}$ denotes a probabilistic polynomial-time (PPT) algorithm that on input 1^λ returns a description $\mathcal{PG} := (\mathbb{G}, \mathbb{G}_T, q, g, e)$ of a symmetric pairing group, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order q for a λ-bit prime q. The group element g is a generator of $\mathbb{G}$. The function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficient computable (non-degenerated) bilinear map (i.e., a pairing). Define $g_T := e(g, g)$, which is a generator in $\mathbb{G}_T$.

We use the implicit representation of group elements as in [21]. For $s \in \{\epsilon, T\}$ and $a \in \mathbb{Z}_q$ define $[a]_s = ag_s \in \mathbb{G}_s$ as the implicit representation of a in $\mathbb{G}_s$. Similarly, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n\times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_s$. Note that it is efficient to compute $[\mathbf{AB}]_s$ given $([\mathbf{A}]_s, \mathbf{B})$ or $(\mathbf{A}, [\mathbf{B}]_s)$ with matching dimensions. Furthermore, $e([\mathbf{A}], [\mathbf{B}]) := [\mathbf{AB}]_T$ can be efficiently computed given $[\mathbf{A}]$ and $[\mathbf{B}]$ with the pairing e.

Many assumptions in paring groups can be expressed as matrix decisional Diffie-Hellman (MDDH) assumption [21].

Definition 2.1 (Matrix distribution). Let $k, \ell \in \mathbb{N}$ with $\ell > k$. We call $\mathcal{D}_{\ell,k}$ a *matrix distribution* if it outputs matrices in $\mathbb{Z}_q^{\ell\times k}$ of full rank k in polynomial time.

For convenience, we define $\mathcal{D}_k := \mathcal{D}_{k+1,k}$. The $\mathcal{D}_{\ell,k}$-matrix Diffie-Hellman problem is to distinguish for $s \in \{\epsilon, T\}$ the two distributions $([\mathbf{A}]_s, [\mathbf{Aw}]_s)$ and $([\mathbf{A}]_s, [\mathbf{u}]_s)$ where $\mathbf{A} \xleftarrow{\$} \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow \mathbb{Z}_q^k$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^\ell$.

Definition 2.2 ($\mathcal{D}_{\ell,k}$-matrix Diffie-Hellman assumption). Let $\mathcal{D}_{\ell,k}$ be a matrix distribution and $s \in \{\epsilon, T\}$. We say that the *$\mathcal{D}_{\ell,k}$-matrix Diffie-Hellman* ($\mathcal{D}_{\ell,k}$-MDDH) *assumption* holds relative to $\mathtt{SymGGen}$ in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{A},\mathcal{D}_{\ell,k},\mathtt{SymGGen},s}(\lambda) := |\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{Aw}]_s) \Rightarrow 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{u}]_s) \Rightarrow 1]|$$

is negligible where the probability is taken over $\mathcal{G} \leftarrow \mathtt{SymGGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_{\ell,k}, \mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^k$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^\ell$.

For $Q \in \mathbb{N}_+$, $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^{k\times Q}, \mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{\ell\times Q}$, consider the Q-fold $\mathcal{D}_{\ell,k}$-MDDH problem which is distinguishing the distributions $(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{AW}]_s)$ and $(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{U}]_s)$. That is, the Q-fold $\mathcal{D}_{\ell,k}$-MDDH problem contains Q independent instances of the $\mathcal{D}_{\ell,k}$-MDDH problem (with the same $\mathbf{A}$ but different $\mathbf{w}_i$). The following lemma gives a reduction from the Q-fold to the standard MDDH assumption with security loss independent of Q.

Lemma 2.3 (Random self-reducibility [21]). *For $\ell > k$ and any matrix distribution $\mathcal{D}_{\ell,k}$, the $\mathcal{D}_{\ell,k}$-MDDH assumption is random self-reducible. In particular, for any $Q \in \mathbb{N}_+$ and any adversary $\mathcal{A}$ there exists an adversary $\mathcal{B}$ with*

$$(\ell - k)\mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B},\mathcal{D}_{\ell,k},\mathtt{SymGGen},s}(\lambda) + \frac{1}{q-1} \geq \mathsf{Adv}^{\mathsf{mddh},Q}_{\mathcal{A},\mathcal{D}_{\ell,k},\mathtt{SymGGen},s}(\lambda) :=$$
$$|\Pr[\mathcal{A}(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{AW}]_s) \Rightarrow 1] - \Pr[\mathcal{A}(\mathcal{PG}, [\mathbf{A}]_s, [\mathbf{U}]_s) \Rightarrow 1]|,$$

where $\mathcal{PG} \leftarrow \texttt{SymGGen}(1^\lambda)$*,* $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}$*,* $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^{k\times Q}$*,* $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{(k+1)\times Q}$*, and* $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathrm{poly}(\lambda)$*, where* poly *is a polynomial independent of* $\mathcal{A}$*.*

In this work we focus on the distribution $\mathcal{U}_{\ell,k}$ of uniformly random full-rank matrices. The following well known Lemma is often helpful with the uniform matrix distribution.

Lemma 2.4

$$\Pr\big[\mathrm{rank}(\mathbf{A}) = k \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{k\times k}\big] \geq 1 - \frac{1}{q-1}$$

Proof. We calculate the probability by successively counting the number of column vectors that form together with n linear independent vectors a linear dependent set (for $n \in \{0, \dots, k-1\}$):

$$\begin{aligned}\frac{\big|\{\mathbf{A} \in \mathbb{Z}_q^{k\times k} \mid \mathrm{rank}(\mathbf{A}) < k\}\big|}{\big|\mathbb{Z}_q^{k\times k}\big|} &= \frac{1}{q^k} + \frac{q}{q^k} + \cdots + \frac{q^{k-1}}{q^k} = \frac{1+q+\cdots+q^{k-1}}{q^k} \\ &= \frac{q^k-1}{q^k(q-1)} = \frac{1}{q-1} - \frac{1}{q^k(q-1)} \leq \frac{1}{q-1}\end{aligned}$$

□

The following Lemma shows that an $\mathcal{U}_{\ell,k}$-MDDH instance is as hard as an $\mathcal{U}_k$-MDDH instance.

Lemma 2.5 ($\mathcal{U}_{\ell,k}$-MDDH $\Leftrightarrow$ $\mathcal{U}_k$-MDDH [23])*. Let* $\ell, k \in \mathbb{N}_+$ *with* $\ell > k$*. More precisely, for each adversary* $\mathcal{A}$ *there exists an adversary* $\mathcal{B}$ *and vice versa with*

$$\mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{A},\mathcal{U}_{\ell,k},\texttt{SymGGen},s}(\lambda) = \mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B},\mathcal{U}_k,\texttt{SymGGen},s}(\lambda)$$

and $T(\mathcal{A}) \approx T(\mathcal{B})$*.*

We use the Q-fold uniform matrix distribution, because for our use case the security loss of $\ell - k$ in Lemma 2.3 is too much and the uniform distribution allows us to give a tight reduction to the standard 1-fold version, as shown by the following Lemma. Gay et al. already provided a tight reduction [23], but their proof is flawed[12] as pointed out by [33].

Lemma 2.6 (Random self-reducibility of $\mathcal{U}_{\ell,k}$-MDDH)*. For every* $\ell > k$ *and every PPT adversary* $\mathcal{A}$ *there exists an adversary* $\mathcal{B}$ *with*

$$\mathsf{Adv}^{\mathsf{mddh},Q}_{\mathcal{A},\mathcal{U}_{\ell,k},\texttt{SymGGen},s}(\lambda) \leq \left\lceil \log\left(\frac{\ell}{k}\right)\right\rceil k\left(\mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B},\mathcal{U}_k,\texttt{SymGGen},s}(\lambda) + \frac{3}{q-1}\right),$$

where $\mathcal{PG} \leftarrow \texttt{SymGGen}(1^\lambda)$ *and* $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathrm{poly}(\lambda)$*, where* poly *is a polynomial independent of* $\mathcal{A}$*.*

Proof. For simplicity, we assume ℓ/k is a power of 2 (if it is not, you can take the next largest power of 2 and cut off the last few entries in the MDDH matrix and in each challenge vector in the end).

We use a hybrid argument with hybrids G_0 to $\mathsf{G}_{\lceil\log(\ell/k)\rceil}$ where in hybrid G_i the adversary gets access to $[\mathbf{A}]_s$ for a matrix $\mathbf{A} \xleftarrow{\$} \mathcal{U}_{\ell,k}$ and Q vectors $[\mathbf{v}_1]_s, \dots, [\mathbf{v}_Q]_s \in \mathbb{G}_s^\ell$ sampled from a uniformly random $k2^i$-dim subspace of $\mathbb{Z}_q^\ell$ that contains the image of $\mathbf{A}$. G corresponds to the real Q-fold $\mathcal{U}_{\ell,k}$-MDDH distribution and $\mathsf{G}_{\lceil\log(\ell/k)\rceil}$ to the randomized Q-fold $\mathcal{U}_{\ell,k}$-MDDH distribution.

The next Lemma describes the transition between the hybrids. Lemma 2.6 follows from the the next Lemma, that proofs the hybrid transition (Lemma 2.7), Lemma 2.3 and Lemma 2.5. □

Lemma 2.7 ($\mathsf{G}_{i-1} \rightsquigarrow \mathsf{G}_i$)*. For every* $i \in \{1, \dots, \lceil\log(\ell/k)\rceil\}$ *and every* $\ell > k$ *and every PPT adversary* $\mathcal{A}$ *there exists an adversary* $\mathcal{B}$ *with*

$$\big|\Pr[\mathsf{G}_{i-1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_i^{\mathcal{A}} \Rightarrow 1]\big| \leq \mathsf{Adv}^{\mathsf{mddh},iQ}_{\mathcal{B},\mathcal{U}_{2k,k},\texttt{SymGGen},s}(\lambda) + \frac{2}{q-1}.$$

[12] They correctly prove that $\mathcal{U}_{\ell,k}$-MDDH is tightly equivalent to $\mathcal{U}_k$-MDDH, but the proof can not show that Q-fold $\mathcal{U}_{\ell,k}$-MDDH is tightly equivalent to Q-fold $\mathcal{U}_k$-MDDH. Thus their argument that this gives a tight reduction when combined with Lemma 2.3 is wrong.

Proof. The reduction chooses 2^i $\ell \times 2k$ matrices $\mathbf{T}_1, \ldots, \mathbf{T}_{2^i}$ uniformly at random. At input of an $2^i Q$-fold $\mathcal{U}_{2k,k}$-MDDH assumption $[\mathbf{A}]_s, [\mathbf{u}_1]_s, \ldots, [\mathbf{u}_{2^i Q}]_s$, the reduction outputs $[\mathbf{B}]_s, [\mathbf{v}_1]_s, \ldots, [\mathbf{v}_Q]_s$ with $\mathbf{B} := \mathbf{T}_1 \mathbf{A}$ and

$$\mathbf{v}_j := \sum_{r=1}^{2^i} \mathbf{T}_r \mathbf{u}_{j+(r-1)Q}.$$

Let us assume that all the column vectors of $(\mathbf{T}_1, \ldots, \mathbf{T}_{2^i})$ are linearly independent. This happens with probability at least $1 - 1/(q-1)$ by Lemma 2.4. In this case these vectors form a uniformly random $k2^i$-fold subspace W of $\mathbb{Z}_q^\ell$.

Next, notice that also the column vectors of $(\mathbf{T}_1\mathbf{A}, \ldots, \mathbf{T}_{2^i}\mathbf{A})$ are distributed uniformly at random (since $\mathbf{A}$ has full rank) and thus they form a uniformly random $k2^i$-fold subspace V of $\mathbb{Z}_q^\ell$ with probability at least $1 - 1/(q-1)$ by Lemma 2.4. In particular, the matrix $\mathbf{B} = \mathbf{T}_1\mathbf{A}$ has full rank.

When the input is a real $2^i Q$-fold $\mathcal{U}_{2k,k}$-MDDH distribution, each vector $\mathbf{u}_s$ for $s \in \{1, \ldots, 2^i Q\}$ can be expressed as $\mathbf{u}_s = \mathbf{A}\mathbf{x}_s$ where $\mathbf{x}_s$ is uniformly at random. Then for each $j \in \{1, \ldots, Q\}$ and the vector $\mathbf{v}_j = \sum_{r=1}^{2^i} \mathbf{T}_r \mathbf{u}_{j+(r-1)Q} = \sum_{r=1}^{2^i} \mathbf{T}_r \mathbf{A} \mathbf{x}_{j+(r-1)Q}$ is a uniformly random vector from V. In this case the reduction simulates G_{i-1}.

When the input is a randomized $2^i Q$-fold $\mathcal{U}_{2k,k}$-MDDH distribution, each vector $\mathbf{u}_s$ is uniformly at random. Then for each $j \in \{1, \ldots, Q\}$ and the vector $\mathbf{v}_j = \sum_{r=1}^{2^i} \mathbf{T}_r \mathbf{u}_{j+(r-1)Q}$ is a uniformly random vector from W. In this case the reduction simulates G_i. □

2.2 Learning With Errors assumption

Gaussian distributions

For a positive definite matrix $\Sigma \in \mathbb{R}^{m \times m}$ there exists a unique positive definite matrix $\sqrt{\Sigma}$ such that $(\sqrt{\Sigma})^2 = \Sigma$. We define the m-dimensional Gaussian function by

$$\begin{aligned} \rho_{\sqrt{\Sigma}} : \mathbb{R}^m &\to (0,1] \\ \mathbf{x} &\mapsto e^{-\pi \mathbf{x}^\top \Sigma^{-1} \mathbf{x}}. \end{aligned}$$

The continuous m-dimensional Gaussian distribution $\mathcal{C}_{\sqrt{\Sigma}}$ is the continuous probability distribution with probability density function $f(\mathbf{x}) = 1/\sqrt{\det(\Sigma)} \rho_{\sqrt{\Sigma}}(\mathbf{x})$. For $\Sigma = \sigma^2 \mathbf{I}_m$, we write $\mathcal{C}_\sigma^m$ instead of $\mathcal{C}_{\sqrt{\Sigma}}$.

The continuous Gaussian distribution is linear, that is, $\mathcal{C}_{\sqrt{\mu^2\Sigma_1 + \Sigma_2}} = \mu \mathcal{C}_{\sqrt{\Sigma_1}} + \mathcal{C}_{\sqrt{\Sigma_2}}$.

The modular Gaussian distribution $\mathcal{C}_{\sqrt{\Sigma},p}$ or $\mathcal{C}_{\sigma,p}$ is obtained by sampling an element according to $\mathcal{C}_{\sqrt{\Sigma}}$ or $\mathcal{C}_\sigma$, respectively, and reducing each entry modulo p.

To avoid calculations with reals, we will use the discrete Gaussian distribution in our scheme. Therefore let Λ be a discrete additive subgroup (a lattice) of $\mathbb{R}^m$ or $\mathbb{R}^m$ modulo a prime p. The discrete Gaussian distribution $\mathcal{D}_{\Lambda,\sigma}$ assigns each vector $\mathbf{x} \in \Lambda$ a probability proportional to $\rho_{\sigma \mathbf{I}_m}(\mathbf{x})$, that is, $\rho_{\sigma \mathbf{I}_m}(\mathbf{x}) / \rho_{\sigma \mathbf{I}_m}(\Lambda)$. The discrete Gaussian distribution is again linear. We will make use of the following tail bound for the discrete Gaussian distribution for $\mathbb{Z}^m$.

Lemma 2.8 ([35, Lemma 4.4]). *For any $k > 1$ we have $\Pr_{\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}}[\|x\| > k\sigma\sqrt{m}] < k^m e^{\frac{m}{2}(1-k^2)}$.*

We will make use of the following lemma to switch between discrete Gaussian and continuous Gaussian distributions. This is a special case of [40, Theorem 1]:

Lemma 2.9 ([40, Theorem 1], [37, Lemma 3.3]). *Let $\sigma, \sigma', \tilde{\sigma} > 0$ and $\varepsilon \in (0,1]$ such that $\sigma^2 = (\sigma')^2 + \tilde{\sigma}^2$ and $\sigma' \geq \sqrt{\ln(2m(1+1/\varepsilon))/\pi}$. Then the distribution that outputs an error vector $\mathbf{e}'' \in \mathbb{Z}^m$ by first sampling $\mathbf{e} \leftarrow \mathcal{C}_{\tilde{\sigma}}$ and then $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^m - \mathbf{e}, \sigma'}$ and finally setting $\mathbf{e}'' := \mathbf{e} + \mathbf{e}'$ is within statistical distance of 8ε of the distribution that directly samples $\mathbf{e}'' \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$.*

Learning with errors assumption

The learning with errors assumption was introduced by [44]. We will use it with subexponential modulus-to-noise ratio to circumvent the difficulties presented in [27].

Definition 2.10 (LWE assumption). The $(n, m, p, \mathcal{E})$-LWE assumption for $n, m, p \in \mathbb{N}_+$ and a distribution $\mathcal{E}$ on $\mathbb{Z}_p^m$, the error distribution, states that for every PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{lwe}}_{n,m,p,\mathcal{E}}(\mathcal{A}) := \Pr[\mathcal{A}(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top) - \mathcal{A}(\mathbf{A}, \mathbf{z})]$$

is negligible in n, where $\mathbf{A} \leftarrow \mathbb{Z}_p^{n \times m}$, $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_p^n$ (for LWE) and $\mathbf{e} \leftarrow \mathcal{E}$.

For the error distribution $\mathcal{E}$ we make the common choice, a discrete Gaussian distribution.

Regev [44] showed that for n growing polynomially in the security parameter and $\mathcal{E} = \mathcal{D}_{\mathbb{Z}_p^n, \sigma}$ with $\sigma > \max\{2\sqrt{n}, q/2^{n^c}\}$ for a constant $c \in (0, 1)$ the hardness of LWE can be reduced quantumly to hard worst case lattice problems, namely approximating the decision version of the shortest vector problem (GapSVP) and the shortest independent vectors problem (SIVP) within $\widetilde{\mathcal{O}}(n2^{n^c})$. Later works also gave classical reductions for $p \geq 2^{n/2}$ consisting of small primes [41] and polynomial moduli [9]. This gives strong indication that the LWE assumption holds for these parameter choices.

Spectral norm of discrete Gaussian

Definition 2.11 (Spectral norm). The spectral norm of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ is defined as

$$\sigma_{\mathbf{A}} := \max_{\mathbf{x} \in \mathbb{R}^m \setminus \{\mathbf{0}\}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}.$$

For matrices $\mathbf{Z} \in \mathbb{Z}_p^{n \times m}$, we define the spectral norm $\sigma_{\mathbf{Z}}$ as the spectral norm of the real valued matrix obtained from $\mathbf{Z}$ by representing each entry in $\{-(p-1)/2, \ldots, (p-1)/2\}$.

The following Lemma states that matrices with discrete Gaussian distributed entries have small spectral norm.

Lemma 2.12 ([8, Proposition 3.1], [36, Lemma 2.8 and 2.9]). *There exists a constant C such that for $m \geq n$ and $\gamma > 0$ the following holds.*

$$\Pr_{\mathbf{Z} \leftarrow \mathcal{D}_{\mathbb{Z}_p^{n \times m}, \gamma}} [\sigma_{\mathbf{Z}} \leq C\gamma\sqrt{m}] \geq 1 - 2^{-m}$$

Decomposition of continuous Gaussian

Lemma 2.13 ([8, Proposition 3.2]). *For a matrix $\mathbf{Z} \in \mathbb{Z}^{n \times m}$, $\sigma_1 > 0$, $\sigma > \sigma_1 \cdot \sigma_{\mathbf{Z}}$ and $\mathbf{\Sigma} := \sigma^2 \mathbf{I}_m - \sigma_1^2 \mathbf{Z}^\top \mathbf{Z}$, the vector $\mathbf{e} = \mathbf{e}_1 \mathbf{Z} + \mathbf{e}_2$ with $\mathbf{e}_1 \leftarrow \mathcal{C}^n_{\sigma_1}$ and $\mathbf{e}_2 \leftarrow \mathcal{C}^m_{\sqrt{\Sigma}}$ is distributed according to $\mathcal{C}^m_\sigma$.*

2.3 Min-entropy

Definition 2.14 (Min-entropy). The min-entropy of a random variable X is defined as

$$H_\infty(X) := -\log(\max_x \Pr[X = x]).$$

Definition 2.15 (Conditional min-entropy). The average min-entropy of a random variable X conditioned on a (correlated) random variable Z is defined as

$$\widetilde{H}_\infty(X \mid Z) := -\log(\mathop{\mathbb{E}}_{z \leftarrow Z}[\max_x \Pr[X = x \mid Z = z]]).$$

Leftover hash Lemma

Definition 2.16 (Statistical distance). The statistical distance between two random variables X and Y is defined as

$$\mathsf{SD}(X, Y) := \frac{1}{2} \sum_x |\Pr[X = x] - \Pr[Y = x]|.$$

Definition 2.17 (Universal hash function). A family $\mathcal{H}$ of functions $h : \mathcal{X} \to \mathcal{Y}$ is called a universal hash family, iff for all $x_1 \neq x_2 \in \mathcal{X}$ we have $\Pr_{h \overset{\$}{\leftarrow} \mathcal{H}}[h(x_1) = h(x_2)] \leq 1/|\mathcal{Y}|$.

Lemma 2.18 (Generalized leftover hash Lemma (GLOHL), [18, Lemma 2.4 in the ePrint version]). *For a random variable X over $\mathcal{X}$, a random variable Z (that can be correlated with X), a function $h : \mathcal{X} \to \mathcal{Y}$ sampled uniformly at random form a universal hash family and $U_{\mathcal{Y}} \xleftarrow{\$} \mathcal{Y}$,*

$$\mathsf{SD}((h(X), h, Z), (U_{\mathcal{Y}}, h, Z)) \leq \frac{1}{2}\sqrt{2^{-\widetilde{H}_\infty(X|Z)}|\mathcal{Y}|}$$

holds.

Noise-lossiness for Gaussians

The following lemma by Brakerski and Döttling gives a lower bound on how much min-entropy of a short secret $\mathbf{s} \in \mathbb{Z}_p^n$ is lost, when continous Gaussian noise is added to $\mathbf{s}$.

Lemma 2.19 ([8, Lemma 5.4]). *Let $\mathbf{s}$ be a random variable on $\{\mathbf{x} \in \mathbb{Z}_p^n \mid \|\mathbf{x}\| \leq r\}$ and $\mathbf{e}_1 \leftarrow \mathcal{C}_{\sigma_1}^n$ for parameters $r, \sigma_1 > 0$. Then the following holds.*

$$\widetilde{H}_\infty(\mathbf{s} \mid \mathbf{s} + \mathbf{e}_1) \geq H_\infty(\mathbf{s}) - \sqrt{2\pi n}\frac{r}{\sigma_1}\log(e).$$

2.4 Non-Interactive Key Exchange

Definition 2.20 (NIKE). A *NIKE* scheme with identity space $\mathcal{IDS}$ and key space $\mathcal{K}$ consists of three polynomial-time algorithms $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey})$, where

- $\mathtt{Setup}$ is a randomized algorithm that takes the unary encoded security parameter 1^λ and samples public parameters pp
- $\mathtt{KeyGen}$ is a randomized algorithm that takes the parameters pp and an identity $\mathsf{id} \in \mathcal{IDS}$ and samples a key pair $(\mathsf{pk}, \mathsf{sk})$
- $\mathtt{SharedKey}$ is a deterministic algorithm that takes the parameters pp, an identity id_1 with its corresponding public key pk_1 and another identity id_2 with its corresponding secret key sk_2 and outputs a shared key K

Definition 2.21 (Correctness). We say that a NIKE $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey})$ for identity space $\mathcal{IDS}$ is *statistically correct*, if the correctness error

$$\sup_{\mathsf{id}_1, \mathsf{id}_2 \in \mathcal{IDS}} \Pr[\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2) \neq \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_2, \mathsf{pk}_2, \mathsf{id}_1, \mathsf{sk}_1) \mid \\ \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id}_2)]$$

is negligible in λ. A NIKE is *perfectly correct* if its correctness error is zero.

The standard security notion for a NIKE is adaptive security. It is a real-or-random notion that allows the adversary to register users, corrupt users, reveal shared keys and get challenged adaptively and arbitrary often. One could strengthen this security notion by giving the adversary an additional oracle that allows him to learn the shared keys of a user and a self-generated public key (dishonest key registration). This security notion can be achieved tightly with little overhead using the generic transformation of [28].

Our first construction in Section 3 achieves a weaker security notion, that we call ν-bounded security, for any $\nu \in \mathbb{N}_{\geq 2}$ with keys that grow linearly in ν. ν-bounded security is defined as adaptive security, but the adversary may only use up to ν users for corruption, revealing shared keys, and challenges. It can still register arbitrary many users and choose adaptively the subset of ν users for the other queries. While this security notion is arguably too weak for most realistic scenarios, it is useful because it implies adaptive security with security loss only $\mathcal{O}((N/\nu)^2)$.

In Section 5 we show how to strengthen our result to achieve ν-semi-adaptive security. This notion is defined like ν-bounded security, except that the adversary can still make $\mathcal{O}_{\mathsf{extr}}$, $\mathcal{O}_{\mathsf{revH}}$ (and $\mathcal{O}_{\mathsf{regH}}$) queries after exceeding the limit of ν involved users. Clearly, ν-semi-adaptive security tightly implies ν-bounded security, but is a more realistic security notion.

```
$\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathtt{NIKE}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{rev}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset;\ \boxed{\dbox{$Q_{\mathsf{inv}} := \emptyset$}}$
    $\mathsf{pks} : \mathcal{IDS} \dashrightarrow \mathcal{PK}$
    $\mathsf{sks} : \mathcal{IDS} \dashrightarrow \mathcal{SK}$
    $b \xleftarrow{\$} \{0,1\}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{revH}}(\cdot,\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$
    if $Q_{\mathsf{rev}} \cap Q_{\mathsf{test}} = \emptyset \wedge \nexists A \in Q_{\mathsf{test}} : A \cap Q_{\mathsf{extr}} = \emptyset$
    $\dbox{$\wedge\ |Q_{\mathsf{inv}}| \leq \nu$}$ then
        return $b \overset{?}{=} b^\star$
    else
        return $0$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}) \neq \bot$ then return $\bot$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id})$
    $\mathsf{pks}(\mathsf{id}) := \mathsf{pk};\ \mathsf{sks}(\mathsf{id}) := \mathsf{sk}$
    return $\mathsf{pk}$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{sks}(\mathsf{id}) \neq \bot$ then
        $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\};\ \boxed{\dbox{$Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$}}$
        return $\mathsf{sks}(\mathsf{id})$
    return $\bot$
```

```
$\mathcal{O}_{\mathsf{revH}}(\mathsf{id}_1 \in \mathcal{IDS}, \mathsf{id}_2 \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1) \neq \bot \wedge \mathsf{sks}(\mathsf{id}_2) \neq \bot$ then
        $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{id}_1, \mathsf{id}_2\}\}$
        $\boxed{\dbox{$Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1, \mathsf{id}_2\}$}}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2)$
        return $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2)$
    return $\bot$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1^\star) \neq \bot \wedge \mathsf{sks}(\mathsf{id}_2^\star) \neq \bot \wedge \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\} \notin Q_{\mathsf{test}}$
    then
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\}$
        $\boxed{\dbox{$Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$}}$
        $\boxed{\textbf{if } |Q_{\mathsf{inv}}| > \nu \textbf{ then return } \bot}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1^\star);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2^\star)$
        $K_0^\star \leftarrow \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1^\star, \mathsf{pk}_1, \mathsf{id}_2^\star, \mathsf{sk}_2)$
        $K_1^\star \xleftarrow{\$} \mathcal{K}$
        return $K_b^\star$
    return $\bot$
```

Figure 2: Experiment for adaptive security, $\boxed{\nu\text{-semi-adaptive security}}$, and $\dbox{$\nu$-bounded security}$ of a NIKE scheme NIKE with identity space $\mathcal{IDS}$ and shared key space $\mathcal{K}$. $\mathcal{PK}$ denotes the public key space and $\mathcal{SK}$ denotes the secret key space. The partial maps pks and sks are initially totally undefined. The set Q_{inv} keeps track of all users involved in the game, that is, users that have been used in at least one $\mathcal{O}_{\mathsf{extr}}$, $\mathcal{O}_{\mathsf{revH}}$ or $\mathcal{O}_{\mathsf{test}}$ query (users that have been only registered but not used since then are not counted as involved users). In the ν-bounded experiment the adversary may involve at most ν users. In the ν-semi-adaptive experiment the adversary may not ask $\mathcal{O}_{\mathsf{test}}$ queries any more after more than ν users have been involved.

Definition 2.22 (Adaptive, ν-bounded, and ν-semi-adaptive security). We say that a NIKE $\mathtt{NIKE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey})$ is ν-bounded, ν-semi-adaptively, or adaptively secure (for $\nu \geq 2$), if for all PPT adversaries $\mathcal{A}$

$$\mathsf{Adv}^{\mathcal{A}\mathsf{xxx}}_{\mathtt{NIKE}}(\lambda) := 2\Pr[\mathsf{Exp}^{\mathsf{xxx}}_{\mathcal{A},\mathtt{NIKE}}(\lambda) \Rightarrow 1] - 1$$

is negligible for $\mathsf{xxx} = \nu\text{-}\mathsf{bounded}$, $\mathsf{xxx} = \nu\text{-}\mathsf{semi\text{-}adaptive}$ or $\mathsf{xxx} = \mathsf{adaptive}$, respectively. The games $\mathsf{Exp}^{\mathsf{xxx}}_{\mathcal{A},\mathtt{NIKE}}(\lambda)$ are defined in Figure 2.

The following argument shows that ν-bounded security implies adaptive security via a non-tight reduction. The reduction forwards the registration queries of up to ν users to the ν-bounded experiment and generates all other keys itself. Then the reduction can randomize the shared keys in the test queries between two users when both of their registrations have been forwarded. Via a hybrid argument, the reduction can randomize all test queries step by step.

Lemma 2.23 *For every NIKE* $\mathtt{NIKE}$ *and every PPT adversary* $\mathcal{A}$ *against the adaptive security of* $\mathtt{NIKE}$*, there exists a PPT adversary* $\mathcal{B}$ *against the* ν*-bounded security for any* $\nu \in \{2, \ldots, N\}$ *with*

$$\mathsf{Adv}^{\mathsf{adaptive}}_{\mathcal{A},\mathtt{NIKE}}(\lambda) \leq \frac{1}{2}\left\lceil \frac{N}{\lfloor \nu/2 \rfloor} + 1 \right\rceil^2 \left(\mathsf{Adv}^{\nu\text{-}\mathsf{bounded}}_{\mathcal{B},\mathtt{NIKE}}(\lambda) + (N_{\mathsf{rev}} + N_{\mathsf{test}})\varepsilon_{\mathtt{NIKE}}(\lambda)\right)$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ *for a polynomial* poly *independent of* $\mathcal{A}$*, where* N *is the maximum number of users that* $\mathcal{A}$ *registers,* N_{rev} *and* N_{test} *are the maximum number of* $\mathcal{A}$*'s* $\mathcal{O}_{\mathsf{revH}}$ *and* $\mathcal{O}_{\mathsf{test}}$ *queries, respectively, and* $\varepsilon_{\mathtt{NIKE}}(\lambda)$ *is the correctness error of* $\mathtt{NIKE}$*.*

Proof. Let $B_1, \ldots, B_{\lceil N/\lfloor \nu/2 \rfloor \rceil}$ be a partition of $\{1, \ldots, N\}$ with $|B_i| \leq \nu/2$ for every $i \in \{1, \ldots, \lceil N/\lfloor \nu/2 \rfloor \rceil\}$ and define

$$\{A_1, \ldots, A_m\} := \{B_i \cup B_j \mid 1 \leq i < j \leq \lceil N/\lfloor \nu/2 \rfloor \rceil\}.$$

The proof uses a hybrid argument with hybrids $\mathsf{G}_0, \ldots, \mathsf{G}_m$ defined in Figure 3. A_i is the set of users whose shared keys will be replaced by random values in a test query in hybrid G_i (if they are not already randomized in G_{i-1}). The construction of these sets ensures that every test query will be randomized in the end, because every pair of users appears in at least one set G_i.

Lemma 2.24 ($\mathsf{G}_{i-1} \rightsquigarrow \mathsf{G}_i$). *For every* $i \in \{1, \ldots, m\}$ *and every PPT adversary* $\mathcal{A}$ *there exists an PPT adversary* $\mathcal{B}$ *such that*

$$\left|\Pr[\mathsf{G}^{\mathcal{A}}_i(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_{i+1}(\lambda) \Rightarrow 1]\right| \leq \mathsf{Adv}^{\nu\text{-}\mathsf{bounded}}_{\mathcal{B},\mathtt{NIKE}}(\lambda) + (N_{\mathsf{rev}} + N_{\mathsf{test}})\varepsilon_{\mathtt{NIKE}}(\lambda).$$

Proof. The reduction generates the public and secret keys for all users of the adaptive experiment directly, except when $\mathsf{ord}(\mathsf{id}) \in A_i$. In this case, the reduction forwards the registration query to the ν-bounded security experiment. $\mathcal{O}_{\mathsf{extr}}$ queries for users with $\mathsf{ord}(\mathsf{id}) \in A_i$, $\mathcal{O}_{\mathsf{revH}}$ queries with $\mathsf{ord}(\mathsf{id}_1), \mathsf{ord}(\mathsf{id}_2) \in A_i$, and $\mathcal{O}_{\mathsf{test}}$ queries with $\mathsf{ord}(\mathsf{id}_1), \mathsf{ord}(\mathsf{id}_2) \in A_i$ and $\nexists j < i : \mathsf{ord}(\mathsf{id}_1), \mathsf{ord}(\mathsf{id}_2) \in A_j$ are also forwarded to the ν-bounded security experiment and the response is directly send to $\mathcal{A}$. The reduction can answer all other queries itself, because it either knows the secret key of at least one of the involved users, or, in $\mathcal{O}_{\mathsf{test}}$ queries with $\exists j < i : \mathsf{ord}(\mathsf{id}_1), \mathsf{ord}(\mathsf{id}_2) \in A_j$, it just has to output a random value. For the $\mathcal{O}_{\mathsf{revH}}$ and $\mathcal{O}_{\mathsf{test}}$ queries it might be necessary to swap the order of the users, but this is not noticeable (with overwhelming probability) due to the correctness of the NIKE. When the ν-bounded security experiment answers the test queries with real keys, the reduction simulates the game G_{i-1}. When the ν-bounded security experiment answers the test queries with random keys, the reduction simulates the game G_i. □

With Lemma 2.24 we can switch from the original game G_0 to G_m and clearly $\Pr[\mathsf{G}^{\mathcal{A}}_m(\lambda) \Rightarrow 1] = 0$, which proves Lemma 2.23. □

For our lower bound on tightness of adaptive NIKE security reductions, we define a relatively weak notion called 2-*step-adaptive security.* The experiment is depicted in Figure 4. It allows the adversary to see $n-2$ secret keys in two loads, and commit to one challenge pair of public keys after seeing the first load. Finally, 2-step-adaptive is the only notion in this paper which is computational, meaning that the adversary has to provide the shared key of the challenge pair in order to win the experiment. To ease presentation of our lower bound proof, the adversary is split into three stateful algorithms $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$.

```
$\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathtt{NIKE}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{rev}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset$
    $\mathsf{pks} : \mathcal{IDS} \dashrightarrow \mathcal{PK}$
    $\mathsf{sks} : \mathcal{IDS} \dashrightarrow \mathcal{SK}$
    $\mathsf{ord} : \mathcal{IDS} \dashrightarrow \{1, \ldots, N\}$
    $b \xleftarrow{\$} \{0,1\}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{revH}}(\cdot,\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$
    if $Q_{\mathsf{rev}} \cap Q_{\mathsf{test}} = \emptyset \wedge \nexists A \in Q_{\mathsf{test}} : A \cap Q_{\mathsf{extr}} = \emptyset$ then
        return $b \overset{?}{=} b^\star$
    else
        return $0$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}) \neq \bot$ then return $\bot$
    Let this be the $j$-th query.
    $\mathsf{ord}(\mathsf{id}) := j$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id})$
    $\mathsf{pks}(\mathsf{id}) := \mathsf{pk};\ \mathsf{sks}(\mathsf{id}) := \mathsf{sk}$
    return $\mathsf{pk}$
```

```
$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{sks}(\mathsf{id}) \neq \bot$ then
        $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\}$
        return $\mathsf{sks}(\mathsf{id})$
    return $\bot$

$\mathcal{O}_{\mathsf{revH}}(\mathsf{id}_1 \in \mathcal{IDS}, \mathsf{id}_2 \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1) \neq \bot \wedge \mathsf{sks}(\mathsf{id}_2) \neq \bot$ then
        $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{id}_1, \mathsf{id}_2\}\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2)$
        return $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2)$
    return $\bot$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1^\star) \neq \bot \wedge \mathsf{sks}(\mathsf{id}_2^\star) \neq \bot \wedge \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\} \notin Q_{\mathsf{test}}$ then
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1^\star);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2^\star)$
        $K_0^\star \leftarrow \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1^\star, \mathsf{pk}_1, \mathsf{id}_2^\star, \mathsf{sk}_2)$
        $K_1^\star \xleftarrow{\$} \mathcal{K}$
        if $\exists j \leq i : \mathsf{ord}(\mathsf{id}_1^\star), \mathsf{ord}(\mathsf{id}_2^\star) \in A_j$ then return $K_1^\star$
        return $K_b^\star$
    return $\bot$
```

Figure 3: Hybrid G_i for the proof of Lemma 2.23. Differences to the original $\mathsf{Exp}^{\mathsf{adaptive}}_{\mathcal{A},\mathtt{NIKE}}(\lambda)$ experiment are highlighted gray. The partial maps pks, sks, ord are initially totally undefined.

```
$\mathsf{Exp}^{\mathsf{2\text{-}step\text{-}adaptive}}_{\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3),N,\mathtt{NIKE}}(\lambda)$:
    $\mathsf{pp} \xleftarrow{\$} \mathtt{NIKE.Setup}(1^\lambda)$
    for $i \in \{1, \ldots, N\}$ do
        $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathtt{NIKE.KeyGen}(\mathsf{pp})$
    $(st_1, D) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$
    $(st_2, \{i^\star, j^\star\}) \leftarrow \mathcal{A}_2(st_1, (\mathsf{sk}_i)_{i \in D})$
    $K^\star \leftarrow \mathcal{A}_3(st_2, (\mathsf{sk}_i)_{i \in [N] \setminus (D \cup \{i^\star, j^\star\})})$
    if $K^\star = \mathtt{NIKE.SharedKey}(\mathsf{pk}_{i^\star}, \mathsf{sk}_{j^\star})$ then
        return $1$
    else
        return $0$
```

Figure 4: Experiment for 2-step-adaptive security of a NIKE scheme $\mathtt{NIKE}$ with shared key space $\mathcal{K}$, for any $N \in \mathbb{N}$. If $i^\star \in D$ or $j^\star \in D$, the experiment aborts.

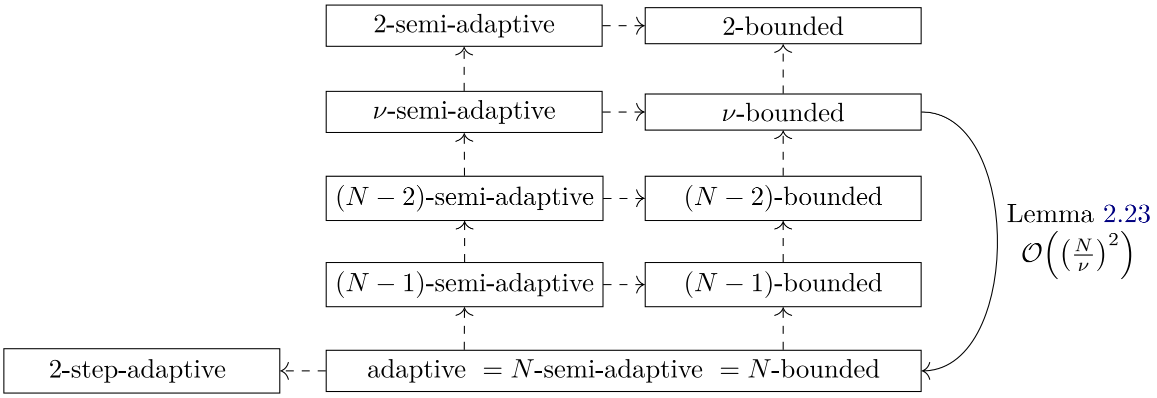

Figure 5: Relations between NIKE security notions for $\nu \in \{2, \ldots, N-2\}$ used in this paper. Dashed arrows mean "tightly implies" and solid arrows mean "implies with specified loss".

```
Setup(1^λ):
    $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \texttt{SymGGen}(1^\lambda)$
    $\mathbf{D} \leftarrow \mathcal{U}_{k+\nu,k}$
    $\mathbf{M} \xleftarrow{\$} \mathsf{Sym}_{k+\nu}(\mathbb{Z}_q)$
    return $\mathsf{pp} := (\mathcal{G}, [\mathbf{D}], [\mathbf{MD}])$

SharedKey(pp, id_1, pk_1, id_2, sk_2):
    return $e(\mathsf{pk}_1^\top, \mathsf{sk}_2)$

KeyGen(pp, id):
    parse $\mathsf{pp} =: (\mathcal{G}, [\mathbf{D}], [\mathbf{MD}])$
    $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^k$
    $\mathsf{pk} := [\mathbf{Dw}]$
    $\mathsf{sk} := [\mathbf{MDw}]$
    return $(\mathsf{pk}, \mathsf{sk})$
```

Figure 6: Our inner-product-based $\texttt{NIKE}_{\mathsf{ip}}$ using symmetric pairing groups.

Definition 2.25 (2-step-adaptive security). We say that a NIKE $\texttt{NIKE} = (\texttt{Setup}, \texttt{KeyGen}, \texttt{SharedKey})$ is 2-step-adaptively secure, if for all PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$

$$\mathsf{Adv}_{\texttt{NIKE}}^{\text{2-step-adaptive}}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3) := \Pr[\mathsf{Exp}_{\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3),N,\texttt{NIKE}}^{\text{2-step-adaptive}}(\lambda) \to 1]$$

is negligible. The experiment is defined in Figure 4.

It is straightforward to verify that 2-step-adaptive security is implied by adaptive security. The relations between the security notions considered in this paper are shown in Figure 5.

3 An inner-product-based NIKE scheme

We present our NIKE $\texttt{NIKE}_{\mathsf{ip}}$ in Figure 6 that tightly achieves ν-bounded security for arbitrary $\nu \geq 2$. However, this comes at the price of public and secret key size $\mathcal{O}(\nu)$. Together with Lemma 2.23, this gives an adaptively secure NIKE with a trade-off between key size and security loss. The security can be based on any MDDH assumption in symmetric pairing groups.

Theorem 3.1 (Correctness). $\texttt{NIKE}_{\mathsf{ip}}$ *is perfectly correct.*

Proof. Let $\mathbf{w}_i$ and $\mathbf{w}_j$ denote the random vectors used to sample the key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ of user id_i and the key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ of user id_j, respectively. Then, using that $\mathbf{M}$ is a symmetric matrix, we get

$$\begin{aligned}\texttt{SharedKey}(\mathsf{pp}, \mathsf{id}_i, \mathsf{pk}_i, \mathsf{id}_j, \mathsf{sk}_j) &= e([(\mathbf{Dw}_i)^\top], [\mathbf{MDw}_j]) = [\mathbf{w}_i^\top \mathbf{D}^\top \mathbf{MDw}_j]_T = [\mathbf{w}_i^\top \mathbf{D}^\top \mathbf{M}^\top \mathbf{Dw}_j]_T \\ &= [\mathbf{w}_j^\top \mathbf{D}^\top \mathbf{MDw}_i]_T = e([(\mathbf{Dw}_j)^\top], [\mathbf{MDw}_i]) = \texttt{SharedKey}(\mathsf{pp}, \mathsf{id}_j, \mathsf{pk}_j, \mathsf{id}_i, \mathsf{sk}_i). \quad \square\end{aligned}$$

```
$\mathsf{G}_0$ $\boxed{\mathsf{G}_1}$

$\mathsf{Exp}^{\nu\text{-bounded}}_{\mathcal{A},\mathtt{NIKE}}(\lambda)$:
    $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathtt{SymGGen}(1^\lambda)$
    $\mathbf{D} \leftarrow \mathcal{U}_{k+\nu,k}$
    $\mathbf{M} \stackrel{\$}{\leftarrow} \mathsf{Sym}_{k+\nu}(\mathbb{Z}_q)$
    $\mathsf{pp} := (\mathcal{G}, [\mathbf{D}], [\mathbf{MD}])$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{rev}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset;\ Q_{\mathsf{inv}} := \emptyset$
    $\mathsf{pks} : \mathcal{IDS} \dashrightarrow \mathcal{PK}$
    $\mathsf{sks} : \mathcal{IDS} \dashrightarrow \mathcal{SK}$
    $b \stackrel{\$}{\leftarrow} \{0,1\}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{revH}}(\cdot,\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$
    if $Q_{\mathsf{rev}} \cap Q_{\mathsf{test}} = \emptyset \wedge \nexists A \in Q_{\mathsf{test}} : A \cap Q_{\mathsf{extr}} = \emptyset \wedge |Q_{\mathsf{inv}}| \leq \nu$ then
        return $b \stackrel{?}{=} b^\star$
    else
        return $0$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_i) \neq \perp$ then return $\perp$
    $\mathbf{w} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^k;\ \mathbf{u} := \mathbf{Dw}$
    $\boxed{\mathbf{u} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{k+\nu}}$
    $\mathsf{pk} := [\mathbf{u}]$
    $\mathsf{sk} := [\mathbf{Mu}]$
    $\mathsf{pks}(\mathsf{id}_i) := \mathsf{pk};\ \mathsf{sks}(\mathsf{id}_i) := \mathsf{sk}$
    return $\mathsf{pk}$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{sks}(\mathsf{id}) \neq \perp$ then
        $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$
        return $\mathsf{sks}(\mathsf{id})$
    return $\perp$

$\mathcal{O}_{\mathsf{revH}}(\mathsf{id}_1 \in \mathcal{IDS}, \mathsf{id}_2 \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1) \neq \perp \wedge \mathsf{sks}(\mathsf{id}_2) \neq \perp$ then
        $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{id}_1, \mathsf{id}_2\}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1, \mathsf{id}_2\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2)$
        return $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2)$
    return $\perp$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1^\star) \neq \perp \wedge \mathsf{sks}(\mathsf{id}_2^\star) \neq \perp \wedge \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\} \notin Q_{\mathsf{test}}$ then
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1^\star);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2^\star)$
        $K_0^\star \leftarrow \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1^\star, \mathsf{pk}_1, \mathsf{id}_2^\star, \mathsf{sk}_2)$
        $K_1^\star \stackrel{\$}{\leftarrow} \mathcal{K}$
        return $K_b^\star$
    return $\perp$
```

Figure 7: Hybrids for the security proof of the NIKE from Figure 6. The partial maps pks and sks are initially totally undefined.

Theorem 3.2 (Security). *For every PPT adversary $\mathcal{A}$ against ν-bounded security of $\mathtt{NIKE}_{\mathsf{ip}}$, there exists a PPT adversary $\mathcal{B}$ solving $\mathcal{U}_k$-MDDH*

$$\mathsf{Adv}^{\nu\text{-bounded}}_{\mathcal{A},\mathtt{NIKE}_{\mathsf{ip}}}(\lambda) \leq \left\lceil \log\left(1 + \frac{\nu}{k}\right)\right\rceil k \left(\mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B},\mathcal{U}_k,\mathtt{SymGGen},s}(\lambda) + \frac{1}{q-1}\right) + \frac{1}{q-1}$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ for a polynomial poly *independent of $\mathcal{A}$.*

The proof uses a hybrid argument with hybrids G_0 and G_1 given in Figure 7.

Lemma 3.3 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). *For every PPT adversary $\mathcal{A}$ there exists an PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq \left\lceil \log\left(1 + \frac{\nu}{k}\right)\right\rceil k \left(\mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B},\mathcal{U}_k,\mathtt{SymGGen},s}(\lambda) + \frac{1}{q-1}\right)$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ for a polynomial poly *independent of $\mathcal{A}$.*

Proof. The real game G_0 uses normal keys (i.e. the public key is a chosen from the linear span of $\mathbf{D}$'s column vectors). In the game G_1 all keys are semi-functional (i.e. the public key is a chosen uniformly at random). Given an N-fold $\mathcal{U}_k$-MDDH challenge $[\mathbf{D}], ([\mathbf{u}_i])_{1\leq i\leq N}$ one can simulate the games G_0 and G_1 efficiently when $\mathbf{A}$ in known over $\mathbb{Z}_q$. If the vectors $\mathbf{u}_i$ are sampled from the linear span of $\mathbf{D}$ this yields the game G_0 and if the vectors $\mathbf{u}_i$ are sampled uniformly random, this yields the game G_1. By reducing the N-fold $\mathcal{U}_k$-MDDH assumption to the $\mathcal{U}_k$-MDDH assumption with Lemma 2.6, the statement follows. □

Lemma 3.4 (G_1). *For every PPT adversary $\mathcal{A}$ there exists an PPT adversary $\mathcal{B}$ such that*

$$|\Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1]| \leq \frac{1}{2} + \frac{1}{q-1}$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\operatorname{poly}(\lambda)$ for a polynomial poly *independent of $\mathcal{A}$.*

Proof. Without loss of generality, assume the adversary involves exactly ν users. Let us assume, that the column vectors of $\mathbf{D}$ and the public keys of all involved users (users with $\mathsf{id} \in Q_{\mathsf{inv}}$ at the end of the game) are linearly independent. Since all the public keys are uniformly random vectors in G_1, this happens with probability at least $1 - 1/(q-1)$ by Lemma 2.4. Initially, the symmetric bilinear form $B(\mathbf{v}, \mathbf{w}) := \mathbf{v}^\top \mathbf{M}\mathbf{w}$ is uniformly random to the adversary. Now suppose the adversary makes a $\mathcal{O}_{\mathsf{test}}$ query with two users $\mathsf{id}_1^\star$ and $\mathsf{id}_2^\star$. Let $[\mathbf{u}_1^\star]$ and $[\mathbf{u}_2^\star]$ be the public keys of $\mathsf{id}_1^\star$ and $\mathsf{id}_2^\star$, respectively and let $[\mathbf{u}_1], \ldots, [\mathbf{u}_{\nu-2}]$ be the public keys of all other users. We show the shared key between the tested users, $[B(\mathbf{u}_1^\star, \mathbf{u}_2^\star)]_T$, is statistically independent of all the other information the adversary learns about B during the game.

Assume all $\mathcal{O}_{\mathsf{revH}}$ queries and other $\mathcal{O}_{\mathsf{test}}$ queries involve at least one involved user different to $\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$, because if the adversary makes a $\mathcal{O}_{\mathsf{revH}}$ query with $\mathsf{id}_1^\star$ and $\mathsf{id}_2^\star$ the adversary has lost trivially and a duplicated $\mathcal{O}_{\mathsf{test}}$ query would only return $\perp$. Thus these queries can not reveal any information about B that is not revealed by $\mathbf{M}(\mathbf{D}|\mathbf{u}_1|\cdots|\mathbf{u}_{\nu-2})$. The public parameters only reveal $\mathbf{MD}$ and any $\mathcal{O}_{\mathsf{extr}}$ query only reveals $\mathbf{M}\mathbf{u}_i$ for an $i \in \{1, \ldots, \nu-2\}$. In total the adversary learns from all queries except the analyzed $\mathcal{O}_{\mathsf{test}}$ query only $\mathbf{M}(\mathbf{D}|\mathbf{u}_1|\cdots|\mathbf{u}_{\nu-2})$. Since the column vectors of $\mathbf{D}$ together with $\{\mathbf{u}_1, \ldots, \mathbf{u}_{\nu-2}, \mathbf{u}_1^\star, \mathbf{u}_2^\star\}$ are assumed to be a linear independent set, $B(\mathbf{u}_1^\star, \mathbf{u}_2^\star) = (\mathbf{u}_1^\star)^\top \mathbf{M}\mathbf{u}_2^\star$ is uniformly random given $\mathbf{M}(\mathbf{D}|\mathbf{u}_1|\cdots|\mathbf{u}_{\nu-2})$. We can apply the above argument to each of the adversaries $\mathcal{O}_{\mathsf{test}}$ queries. Consequently, the adversaries advantage in G_1 is 0 under the stated assumptions. □

Proof (of Theorem 3.2). Combining Lemmata 3.3 and 3.4 proves Theorem 3.2. □

Corollary 3.5 *The NIKE* $\texttt{NIKE}_{\mathsf{ip}}$ *is adaptively secure with a security loss of $\mathcal{O}((N/\nu)^2 \log \nu)$ under the decision linear (DLIN) assumption.*

Proof. The DLIN assumption implies the $\mathcal{U}_k$-MDDH for $k \geq 2$ [21], so we can set $k = 2$. The NIKE then achieves ν-bounded security with loss $\mathcal{O}(\log \nu)$ by Theorem 3.2. With Lemma 2.23 we can achieve adaptive security by increasing the security loss by a factor of $\mathcal{O}((N/\nu)^2)$. □

4 Lower bound

In this section, we show that for all NIKEs that follow a special structure, there is an in some sense inherent trade-off between key sizes and quality of the reduction. To that end, and inspired by our $\texttt{NIKE}_{\mathsf{ip}}$ presented in the previous section, we introduce the notion of an "inner-product NIKE", which basically requires that the shared keys can be expressed as the inner product of information from the public-key and the secret-key. Even though this might seem a bit artificial, we will show that this indeed captures not only $\texttt{NIKE}_{\mathsf{ip}}$, but a large class of DH-based NIKEs. The inner-product structure together with the strong structural requirement of the NIKE that shared keys have to be consistent will allow us to show that at some point (i.e. after sufficiently many secret keys are fixed) the shared keys between *all* key pairs are uniquely determined. Therefore, we can argue that at this point for each remaining key pair the reduction either knows the unique shared key (and therefore an external adversary is not helpful) or it does not know the shared key and therefore has to abort whenever the adversary asks for the secret key of either of the involved parties. Compared to previous lower bounds on NIKE reductions [2, 28], we do not make the generic assumption that pairs of public keys already determine the corresponding shared key. Instead, we leave room for a reduction to adaptively determine secret keys upon corruptions, as long as they follow the inner product structure we require. We start this section by recalling some preliminaries on how to formalize non-interactive complexity assumptions and simple reductions. We then introduce the notion of inner-product NIKE and show that our $\texttt{NIKE}_{\mathsf{ip}}$ is an inner-product NIKE. Further, we show that the structure of inner product NIKE forces uniqueness of shared keys after giving out sufficiently many secret keys. Using this, we show a lower bound on inner-product NIKEs, given a trade-off between the tightness of the reduction and the dimension of the inner-product NIKE.

```
$\mathsf{Exp}^{\mathsf{nica}}_{\mathcal{B},\mathcal{N}=(G,U,V)}(\lambda)$:
    $(c, w) \xleftarrow{\$} G(1^\lambda)$
    $s \leftarrow \mathcal{B}(c)$
    return $V(c, w, s)$
```

Figure 8: Security experiment for a non-interactive complexity assumption (NICA).

```
$\Lambda^{\mathcal{A}}(c; r_\Lambda \xleftarrow{\$} \{0,1\}^\lambda)$:
    $(\mathsf{pp}, \mathsf{pk}_1, \dots, \mathsf{pk}_N, st_{\Lambda_1}) \leftarrow \Lambda_1(c, r_\Lambda)$
    $(st_1, D) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk}_1, \dots, \mathsf{pk}_N)$
    $((\mathsf{sk}_i)_{i \in D}, st_{\Lambda_2}) \leftarrow \Lambda_2(D, st_{\Lambda_1})$
    $(st_2, \{i^*, j^*\}) \leftarrow \mathcal{A}_2(st_1, (\mathsf{sk}_i)_{i \in D})$
    $((\mathsf{sk}_i)_{i \in [N] \setminus (D \cup \{i^\star, j^\star\})}, st_{\Lambda_3}) \leftarrow \Lambda_3(\{i^*, j^*\}, st_{\Lambda_2})$
    $(K^*) \leftarrow \mathcal{A}_3(st_2, (\mathsf{sk}_i)_{i \in [N] \setminus (D \cup \{i^\star, j^\star\})})$
    $s \leftarrow \Lambda_4(K^*, st_{\Lambda_3})$
    return $s$
```

Figure 9: The simple reduction $\Lambda^{\mathcal{A}}$

4.1 Preliminaries on meta-reduction techniques

We recall the definition of a non-interactive complexity assumption from [2], Def. 4 and 5. A non-interactive complexity assumption consists of an instance generation algorithm, the "trivial" solver to be able to define the advantage over the trivial strategy (for instance, the algorithm that simply outputs a guess for decision assumptions), and the verification algorithm, that verifies whether a given solution was correct.

Definition 4.1 (Non-interactive complexity assumption). A *non-interactive complexity assumption (NICA)* $\mathcal{N} = (G, U, V)$ consists of three turing machines. The instance generation machine $(c, w) \xleftarrow{\$} G(1^\lambda)$ takes the security parameter as input, and outputs a problem instance c and a witness w. U is a PPT machine, which takes as input c and outputs a candidate solution s. The verification TM V takes as input (c, w) and a candidate solution s. If $V(c, w, s) = 1$, then we say that s is a correct solution to the challenge c.

Definition 4.2 We say that a PPT algorithm $\mathcal{B}$ ε-breaks a NICA $\mathcal{N} = (G, U, V)$ if it holds that

$$\mathsf{Adv}^{\mathsf{nica}}_{\mathcal{B},\mathcal{N}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{nica}}_{\mathcal{B},\mathcal{N}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{Exp}^{\mathsf{nica}}_{U,\mathcal{N}}(\lambda) \Rightarrow 1]| \geq \varepsilon(\lambda),$$

where $\mathsf{Exp}^{\mathsf{nica}}_{\mathcal{B},\mathcal{N}=(G,U,V)}$ is the experiment defined in Figure 8 and the probability is taken over the random coins consumed by G and the uniformly random choices in the experiment.

Next, we recall "simple reductions" [2]. Basically, a simple reduction is a reduction that uses the adversary in a *black-box way* and *only once without rewinding.* While this might seem like a very strong requirement on the reduction, actually many cryptographic reductions satisfy it.

Definition 4.3 (Simple Reduction). Let $\mathcal{N} = (G, U, V)$ be a NICA and `NIKE` an NIKE, where the number of users is denoted by N. A PPT algorithm $\Lambda := (\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4)$ is a *simple* $(\varepsilon_\Lambda, \varepsilon_{\mathcal{A}})$*-reduction* from breaking $\mathcal{N} = (G, U, V)$ to breaking the N-user 2-step-adaptive-security of `NIKE`, if for any PPT algorithm $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that $\varepsilon_{\mathcal{A}}$-breaks the N-user 2-step-adaptive-security of `NIKE`, $\Lambda^{\mathcal{A}}$ runs in time $T(\Lambda) + T(\mathcal{A})$ and ε_Λ-breaks $\mathcal{N} = (G, U, V)$, where $\Lambda^{\mathcal{A}}$ is defined in Figure 9.

4.2 Inner-product NIKE

We now formalize the notion of *inner-product NIKE.* The intuition behind this definition is as follows. Basically, we require an (inefficient) algorithms `Extract` that can be used to extract the key vectors $(\mathbf{x}, \mathbf{y})$

from a valid pair of public and secret keys, such that $(\mathbf{x},\mathbf{y})$ can be used to compute the shared key as an inner product. As we will see in the next section, this inner-product structure will enforce uniqueness of the shared keys, as soon as sufficiently many secret keys are fixed. The verification algorithm is necessary to ensure that the reduction can only give out public and secret keys that satisfy some structural requirements (e.g. be of the right form and dimension). The public extraction algorithm $\mathtt{PExtract}$ together with the *binding* requirement is necessary to ensure that the public keys are committing to the vector $\mathbf{x}$, even before the secret keys are given out. Finally, we therefore require the function f to be invertible, since it will be crucial that there is a one-to-one correspondence between the inner product and the shared key algorithm (note though that the inverse does not have to be efficiently computable).

Definition 4.4 (Inner-product NIKE). Let $p \in \mathbb{N}$ a prime. We say a NIKE $\mathtt{NIKE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey})$ is a d-dimensional *inner-product NIKE (ip-NIKE)* over $\mathbb{Z}_p$, if there exists:

- a PPT algorithm $\mathtt{Ver}$ taking as input public parameters pp, and a key pair $(\mathsf{pk},\mathsf{sk})$ and returning a bit $b \in \{0,1\}$,
- an (inefficient) deterministic extractor $\mathtt{Extract}$ that takes as input public parameters pp, and a key pair $(\mathsf{pk},\mathsf{sk})$ and returns a tuple $(\mathbf{x},\mathbf{y}) \in \mathbb{Z}_p^d \times \mathbb{Z}_p^d$,
- an (inefficient) deterministic extractor $\mathtt{PExtract}$ taking as input pp and pk and returning a vector $\mathbf{x} \in \mathbb{Z}_p^d$,
- and a function f taking as input public parameters pp and an element $z \in \mathbb{Z}_p$ and returning a element in the image of $\mathtt{NIKE.SharedKey}$.

such that the following properties hold.

(i) *Verifiable keys.* For all $(\mathsf{pk},\mathsf{sk})$ in the image of $\mathtt{NIKE.KeyGen}(\mathsf{pp})$ it holds $\mathtt{Ver}(\mathsf{pp},\mathsf{pk},\mathsf{sk}) = 1$.

(ii) *Ip-computable shared keys.* For all public parameters pp, and all key pairs $(\mathsf{pk},\mathsf{sk}), (\mathsf{pk}',\mathsf{sk}')$ with $\mathtt{Ver}(\mathsf{pp},\mathsf{pk},\mathsf{sk}) = \mathtt{Ver}(\mathsf{pp},\mathsf{pk}',\mathsf{sk}') = 1$, for $(\mathbf{x},\mathbf{y}) \leftarrow \mathtt{Extract}(\mathsf{pp},\mathsf{pk},\mathsf{sk})$ and $(\mathbf{x}',\mathbf{y}') \leftarrow \mathtt{Extract}(\mathsf{pp}, \mathsf{pk}',\mathsf{sk}')$ it holds
$$\mathtt{SharedKey}(\mathsf{pp},\mathsf{pk},\mathsf{sk}') = f(\mathsf{pp}, \langle \mathbf{x},\mathbf{y}' \rangle).$$

(iii) *Binding public keys.* For all $(\mathsf{pk},\mathsf{sk})$ with $\mathtt{Ver}(\mathsf{pk},\mathsf{sk}) = 1$ for $\mathbf{x} \leftarrow \mathtt{PExtract}(\mathsf{pp},\mathsf{pk})$ and $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \leftarrow \mathtt{Extract}(\mathsf{pp},\mathsf{pk},\mathsf{sk})$ it holds $\mathbf{x} = \tilde{\mathbf{x}}$.

(iv) *Invertibility of f.* The induced function $f_{\mathsf{pp}} = f(\mathsf{pp},\cdot)$ is injective with inverse f_{pp}^{-1}.

We call a key pair $(\mathsf{pk},\mathsf{sk})$ with $\mathtt{Ver}(\mathsf{pp},\mathsf{pk},\mathsf{sk}) = 1$ *valid.* By the *ip-dimension* of an inner-product NIKE $\mathtt{NIKE}$, we denote the minimal dimension d, such that $\mathtt{NIKE}$ satisfies the definition of d-dimensional inner-product NIKE.

Example: $\mathtt{NIKE}_{\mathsf{ip}}$ is an ip-NIKE. While at first sight Definition 4.4 of an ip-NIKE, for which we prove our lower bound on tightness, seems artificial, it in fact covers most DH-based NIKE schemes from the literature. In the following, we show that our $\mathtt{NIKE}_{\mathsf{ip}}$ from Section 3 in fact meets the definition. For details on why other types of common NIKEs such as the Diffie-Hellman key exchange fit our framework, we refer to Appendix A.

Recall that $\mathtt{NIKE}_{\mathsf{ip}}$ (Figure 6) is of the following form: $\mathsf{pp} = (\mathcal{G}, [\mathbf{D}], [\mathbf{MD}])$, where $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, q, P, e)$, $\mathbf{D} \in \mathbb{Z}_q^{(k+\nu)\times k}$ and $\mathbf{M} \in \mathbb{Z}_q^{(k+\nu)\times(k+\nu)}$ a symmetric matrix. Further, we have $\mathsf{pk} = [\mathbf{Dw}]$, $\mathsf{sk} = [\mathbf{MDw}]$ for $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^k$ and $\mathtt{SharedKey}(\mathsf{pp},\mathsf{pk},\mathsf{sk}') = e(\mathsf{pk}^\top, \mathsf{sk}')$. $\mathtt{NIKE}_{\mathsf{ip}}$ is an ip-NIKE together with the following algorithms:

- $\mathtt{Ver}$ is the algorithm that returns 1 if pk,sk are both elements of $\mathbb{G}^{k+\nu}$,
- $\mathtt{Extract}$ is the algorithm that on input of a tuple $(\mathsf{pk},\mathsf{sk}) \in \mathbb{G}^{k+\nu} \times \mathbb{G}^{k+\nu}$ outputs the tuple of vectors $(\mathbf{x},\mathbf{y}) \in \mathbb{Z}_q^{k+\nu} \times \mathbb{Z}_q^{k+\nu}$ such that $[\mathbf{x}] = \mathsf{pk}$ and $[\mathbf{y}] = \mathsf{sk}$
- $\mathtt{PExtract}$ is the algorithm that on input of an element $\mathsf{pk} \in \mathbb{G}^{k+\nu}$ returns the vector $\mathbf{x} \in \mathbb{Z}_q^{k+\nu}$ such that $[\mathbf{x}] = \mathsf{pk}$.

- and f via $f_{\mathsf{pp}}\colon \mathbb{Z}_q \to \mathbb{G}_T, f_{\mathsf{pp}}(z) := [z]_T$.

Then, we have

(i.) *Verifiable keys.* This is immediate, since $\mathtt{KeyGen}$ always returns tuples in $\mathbb{G}^{k+\nu} \times \mathbb{G}^{k+\nu}$. We intentionally expand the set of key pairs accepted by $\mathtt{Ver}$ to the full group, to capture reductions that produce key pairs in $\mathbb{G}^{k+\nu} \times \mathbb{G}^{k+\nu} \setminus \mathsf{Image}(\mathtt{KeyGen})$.

(ii.) *Ip-computable shared keys.* For all valid key pairs $(\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}', \mathsf{sk}')$ we have

$$\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}') = e(\mathsf{pk}^\top, \mathsf{sk}') = e([\mathbf{x}^\top], [\mathbf{y}']) = [\mathbf{x}^\top \mathbf{y}']_T = f_{\mathsf{pp}}(\langle \mathbf{x}, \mathbf{y}' \rangle),$$

where $(\mathbf{x}, \mathbf{y}) \leftarrow \mathtt{Extract}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$ and $(\mathbf{x}', \mathbf{y}') \leftarrow \mathtt{Extract}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}')$.

(iii.) *Binding public keys.* This is immediate by construction of $\mathtt{Extract}$ and $\mathtt{PExtract}$, since P is a generator of $\mathbb{G}$.

(iv.) *Invertibility of f.* Since $e(P, P)$ generates $\mathbb{G}_T$, f_{pp} is invertible.

Limitations of the notion of ip-NIKE. Lattice-based NIKEs (as described e.g. in [27]) are not captured by the notion of ip-NIKE, because the Rounding function that is applied to obtain the shared key is not injective.

4.3 Lower bound for inner-product NIKEs

In order to show our lower bound, we first prove that after giving out sufficiently many public key/secret key pairs for a NIKE that satisfies the inner-product form, the reduction is committed to all shared keys. More precisely, let $\{\mathsf{pk}_i\}_{i\in I}$ such that the corresponding vectors $\mathbf{x}_i \leftarrow \mathtt{PExtract}(\mathsf{pp}, \mathsf{pk}_i)$ span the whole space $\mathbb{Z}_p^d$.[13] We further fix secret keys for pk_i for all $i \in I$ such that $(\mathsf{pk}_i, \mathsf{sk}_i)$ are all valid and *consistent* with each other, meaning that for all i, j, we have $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_j) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_j, \mathsf{sk}_i)$. We will show that such a set of keys fix not only the shared keys between the public keys within $\{\mathsf{pk}_i\}_{i\in I}$, but the shared keys between *all* possible valid public keys in the system. Therefore, if d is significantly smaller than N, then with high probability, any large enough random subset of key pairs will span the whole space of public keys and thereby already fix all shared keys (computed as inner products) in the system. This will be crucial in our meta reduction, where we use uniqueness of shared keys in order to efficiently simulate a hypothetical perfect adversary, thereby essentially showing that the reduction either has to abort with large probability or would be able to solve the underlying problem itself.

Lemma 4.5 (Unique shared keys for ip-NIKEs). *Let $d, p, \lambda \in \mathbb{N}$ and $\mathtt{NIKE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey}, \mathtt{Ver}, \mathtt{Extract}, \mathtt{PExtract})$ a d-dimensional ip-NIKE over $\mathbb{Z}_p$. Let $I \subset [N]$. Let $\mathsf{pp} \in \{0,1\}^\star$ and let $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in I}, (\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}', \mathsf{sk}')$ be such that:*

(a) All key pairs are valid: $\mathtt{Ver}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i) = 1$ *for all* $i \in I$ *and* $\mathtt{Ver}(\mathsf{pp}, (\mathsf{pk}, \mathsf{sk})) = \mathtt{Ver}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}') = 1$.

(b) The key pairs in I are pairwise consistent:
$\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_j) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_j, \mathsf{sk}_i)$ *for all* $i, j \in I$ *with* $i \neq j$.

(c) The key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}', \mathsf{sk}')$ are consistent with the key pairs in I: $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}_i) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk})$ *and* $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}_i) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}')$ *for all* $i \in I$.

(d) The public keys $\mathsf{pk}, \mathsf{pk}'$ are "in the span" of $\{\mathsf{pk}_i\}_{i\in I}$: *for* $\mathbf{x}_i \leftarrow \mathtt{PExtract}(\mathsf{pp}, \mathsf{pk}_i)$ *and* $\mathbf{x} \leftarrow \mathtt{PExtract}(\mathsf{pp}, \mathsf{pk})$*,* $\mathbf{x}' \leftarrow \mathtt{PExtract}(\mathsf{pp}, \mathsf{pk}')$ *it holds that* $\mathbf{x}, \mathbf{x}'$ *are in the span of* $\{\mathbf{x}_i\}_{i\in I}$.

Then it holds that $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}') = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk})$*. In other words, the shared key between the users holding* pk' *and* pk *is* consistent*.*

[13] For simplicity of this explanation we assume for now that such a set of keys exists, but stress that our results do not rely on it.

Proof. Let $(\mathbf{x}_i, \mathbf{y}_i) \leftarrow \mathtt{Extract}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i)$, $(\mathbf{x}, \mathbf{y}) \leftarrow \mathtt{Extract}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$, and $\mathbf{x}' \leftarrow \mathtt{PExtract}(\mathsf{pp}, \mathsf{pk}')$. Note that $\mathbf{x}$ can be extracted from pk independently of the corresponding secret key due to the "binding public keys" property Def. 4.4 (iii). We can rely on the latter property since key pairs are valid because of condition (a). Again because of (a), shared keys are ip-computable, and we have

$$\begin{aligned} \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}') &= \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}) \\ \overset{\text{Def. 4.4 (ii)}}{\Longleftrightarrow} \quad f_{\mathsf{pp}}(\langle \mathbf{x}, \mathbf{y}' \rangle) &= f_{\mathsf{pp}}(\langle \mathbf{x}', \mathbf{y} \rangle) \\ \overset{f_{\mathsf{pp}} \text{ invertible}}{\Longleftrightarrow} \quad \langle \mathbf{x}, \mathbf{y}' \rangle &= \langle \mathbf{x}', \mathbf{y} \rangle, \end{aligned}$$

and thus it suffices to show equality of these inner products. Due to validity of all involved key pairs and condition (b), for all $i, j \in I$ with $i \neq j$, it holds:

$$\begin{aligned} \langle \mathbf{x}_i, \mathbf{y}_j \rangle &\overset{\text{Def. 4.4 (ii)}}{=} f_{\mathsf{pp}}^{-1}(\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_j)) \\ &\overset{\text{Cond. (b)}}{=} f_{\mathsf{pp}}^{-1}(\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_j, \mathsf{sk}_i)) \overset{\text{Def. 4.4 (ii)}}{=} \langle \mathbf{x}_j, \mathbf{y}_i \rangle. \end{aligned} \tag{1}$$

Analogously, by to validity of all involved key pairs and condition (c) for all $i \in I$, we have:

$$\langle \mathbf{x}_i, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y}_i \rangle \qquad \text{and} \qquad \langle \mathbf{x}_i, \mathbf{y}' \rangle = \langle \mathbf{x}', \mathbf{y}_i \rangle. \tag{2}$$

By condition (d) from the lemma statement, we can find $\beta, \gamma \in \mathbb{Z}_p^{|I|}$ with $\sum_{j=1}^{|I|} \beta_j \mathbf{x}_j = \mathbf{x}$ and $\sum_{i=1}^{|I|} \gamma_i \mathbf{x}_i = \mathbf{x}'$. First, note that for all $i \in I$ we have

$$\langle \mathbf{x}_i, \sum_{j=1}^{|I|} \beta_j \mathbf{y}_j \rangle = \sum_{j=1}^{|I|} \beta_j \langle \mathbf{x}_i, \mathbf{y}_j \rangle \overset{\text{Eq. 1}}{=} \sum_{j=1}^{|I|} \beta_j \langle \mathbf{x}_j, \mathbf{y}_i \rangle = \langle \mathbf{x}, \mathbf{y}_i \rangle \overset{\text{Eq. 2}}{=} \langle \mathbf{x}_i, \mathbf{y} \rangle. \tag{3}$$

With this, it follows that

$$\langle \mathbf{x}, \mathbf{y}' \rangle \overset{\text{Cond. (d)}}{=} \langle \sum_{j=1}^{|I|} \beta_j \mathbf{x}_j, \mathbf{y}' \rangle = \sum_{j=1}^{|I|} \beta_j \langle \mathbf{x}_j, \mathbf{y}' \rangle \overset{\text{Eq. 2}}{=} \sum_{j=1}^{|I|} \beta_j \langle \mathbf{x}', \mathbf{y}_j \rangle = \langle \mathbf{x}', \sum_{j=1}^{|I|} \beta_j \mathbf{y}_j \rangle.$$

Finally, we have

$$\langle \mathbf{x}', \sum_{j=1}^{|I|} \beta_j \mathbf{y}_j \rangle = \langle \sum_{i=1}^{|I|} \gamma_i \mathbf{x}_i, \sum_{j=1}^{|I|} \beta_j \mathbf{y}_j \rangle = \sum_{i=1}^{|I|} \gamma_i \langle \mathbf{x}_i, \sum_{j=1}^{|I|} \beta_j \mathbf{y}_j \rangle \overset{\text{Eq. 3}}{=} \sum_{i=1}^{|I|} \gamma_i \langle \mathbf{x}_i, \mathbf{y} \rangle = \langle \mathbf{x}', \mathbf{y} \rangle,$$

which concludes the proof. □

Note that this in particular implies that the shared key is *independent of the choice of secret keys* $\mathsf{sk}, \mathsf{sk}'$ *satisfying conditions (a) and (c)*, as captured in the following corollary.

Corollary 4.6 *Let* $\mathsf{pp} \in \{0,1\}^\star$ *and let* $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in I}, (\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}', \mathsf{sk}')$ *and* $\hat{\mathsf{sk}}$ *such that* $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in I}$, $(\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}', \mathsf{sk}')$ *as well as* $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in I}, (\mathsf{pk}, \hat{\mathsf{sk}}), (\mathsf{pk}', \mathsf{sk}')$ *satisfy conditions (a) to (d) in Lemma 4.5. Then we have* $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}', \hat{\mathsf{sk}})$.

Proof. By Lemma 4.5 we have $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}')$ as well as $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}', \hat{\mathsf{sk}}) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}')$, the corollary is thus immediate. □

Relying on the fact that after giving out sufficiently many secret keys, all shared keys are uniquely determined, we are able to prove a trade-off between the tightness of the reduction and the dimension of an inner-product NIKE. We formalize this in the following theorem and give an interpretation of our result below.

Theorem 4.7 *Let $\mathcal{N} = (G, U, V)$ be a non-interactive complexity assumption, let $N, d \in \mathbb{N}$ with $4d+6 < N$, and let $p \in \mathbb{N}$ a prime. Let* `NIKE` *be a perfectly correct 2-step-adaptively-secure d-dimensional ip-NIKE over $\mathbb{Z}_p$ with shared key space $\mathcal{K}$, public key space $\mathcal{PK}$ and secret key space $\mathcal{SK}$. Then, for any simple $(\varepsilon_\Lambda, \varepsilon_\mathcal{A})$-reduction from breaking the NICA $\mathcal{N}$ to breaking the N-user 2-step-adaptive security of* `NIKE`*, there exists a PPT adversary $\mathcal{B}$ on the NICA $\mathcal{N}$, such that*

$$\varepsilon_\Lambda \leq \frac{4d+6}{N} \cdot \varepsilon_\mathcal{A} + \mathsf{Adv}^{\mathsf{nica}}_{\mathcal{N},\mathcal{B}}.$$

Proof. To simplify presentation we will assume that the number of users N is even. We follow the proof structure of [30], [34], [2]. The idea of the meta-reduction is as follows. We will first describe a hypothetical adversary on the NIKE that has winning probability $\varepsilon_\mathcal{A}$ in the real game. This adversary simply brute-forces secret keys that are consistent with the key pairs given out by the experiment and uses those to compute the shared key, which it then outputs with probability $\varepsilon_\mathcal{A}$. (We will of need to show that this shared key is uniquely determined after a sufficient number of corruptions.) This on its own is of course not helpful, because the adversary will not be efficient, but the idea of the meta reduction is now to show how to efficiently simulate this adversary by rewinding the reduction. We will show that we can actually *perfectly* simulate the hypothetical adversary, except with probability $(4d+6)/N$. What is crucial in this argument is that the shared key computed by the hypothetical adversary is *unique* after sufficiently many secret keys are fixed. For this, we rely on Lemma 4.5 and Corollary 4.6.

The hypothetical adversary. In the following, we describe a hypothetical (but inefficient) 2-step-adaptive adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ with success probability $\varepsilon_\mathcal{A}$. The adversary starts with tossing a coin χ with $\Pr[\chi = 1] = \varepsilon_\mathcal{A}$. If $\chi = 0$, $\mathcal{A}$ sets $\mathsf{out}_\mathcal{A} = \bot$.

$\mathcal{A}_1(\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ first extracts $\mathbf{x}_i \leftarrow \mathtt{PExtract}(\mathsf{pp}, \mathsf{pk}_i)$ for all $i \in [N]$. Then, $\mathcal{A}$ first samples a set $D^\star$ of size $N/2$. Then, samples challenge indices $i^\star, j^\star \xleftarrow{\$} D^\star$ with $i^\star \neq j^\star$ until $D = D^\star \backslash \{i^\star, j^\star\}$ is such that $\mathbf{x}_{i^\star}, \mathbf{x}_{j^\star}$ are in the span of $\{\mathbf{x}_i\}_{i \in D}$. (Since $N/2 > d+2$, this will succeed eventually for any choice of $D^\star$.) Once successful, it outputs $st_1 = (\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, \mathbf{x}_1, \ldots, \mathbf{x}_N, i^\star, j^\star, D)$ and D to the experiment.

$\mathcal{A}_2(st_1, (\mathsf{sk}_i)_{i \in D})$ checks whether $\mathtt{Ver}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i) = 1$ for all $i \in D$. It further checks if $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_j, \mathsf{sk}_i) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_j)$ for all $i, j \in D$. If either check fails, it sets $\mathsf{out}_\mathcal{A} = \bot$. Next, it outputs $st_2 = (st_1, (\mathsf{sk}_i)_{i \in D})$ and $(i^\star, j^\star)$ to the experiment.

$\mathcal{A}_3(st_2, (\mathsf{sk}_i)_{i \in [N] \backslash (D \cup \{i^\star, j^\star\})})$ checks whether for all $i \in [N] \backslash (D \cup \{i^\star, j^\star\})$ it holds $\mathtt{Ver}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i) = 1$. It further checks if $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_j) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_j, \mathsf{sk}_i)$ for all $i \neq j \in D \cup C$, and sets $\mathsf{out}_\mathcal{A} = \bot$ if one of the conditions is not met. Finally, it computes $\mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$ such that $\mathtt{Ver}(\mathsf{pk}_{i^\star}, \mathsf{sk}_{i^\star}) = \mathtt{Ver}(\mathsf{pk}_{j^\star}, \mathsf{sk}_{j^\star}) = 1$ and such that $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_{i^\star}, \mathsf{sk}_i) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_{i^\star})$ and $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_{j^\star}, \mathsf{sk}_i) = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_{j^\star})$ for all $i \in D$, and sets $\mathsf{out}_\mathcal{A} = \bot$ if no such secret keys exist. (Note that the hypothetical adversary does *not* check consistency with the key pairs in $[N] \backslash (D \cup \{i^\star, j^\star\})$.) If $\mathsf{out}_\mathcal{A} = \bot$, the adversary $\mathcal{A}$ outputs $\bot$. Otherwise, it returns $K^\star = \mathtt{SharedKey}(\mathsf{pk}_{i^\star}, \mathsf{sk}_{j^\star})$ to the experiment.

Success probability of $\mathcal{A}$. By Lemma 4.5 and Corollary 4.6 it follows that if $\mathcal{A}$ does not abort, the shared key $K^\star$ is unique and independent of the choice of $\mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$ computed by the hypothetical adversary, since we are guaranteed that $\mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$ are valid and consistent with all key pairs in D (which again are valid and consistent which each other). We can therefore conclude that the shared key computed by the hypothetical adversary has to be the same as the one computed by the honest 2-step-adaptive challenger, where we are guaranteed consistency of all key pairs by the perfect correctness of the NIKE, and validity because by definition of `Ver` all key pairs computed by `KeyGen` are valid. Therefore, $\mathcal{A}$ breaks 2-step-adaptive security of `NIKE` with probability $\varepsilon_\mathcal{A}$.

Simulating the hypothetical adversary. We now describe an attacker $\mathcal{B}$ on the underlying NICA $\mathcal{N}$ that builds on the reduction Λ. To that end, we describe how to efficiently simulate the adversary $\mathcal{A}$. We start with an informal description to illustrate the main ideas, and then proceed to a formal description.

The simulated adversary cannot extract the vectors $\mathbf{x}$ from the public keys, because `PExtract` will in general not be efficient. Instead, the adversary simply samples a random set D and "hopes" that this set is indeed spanning the space containing the vectors $\texttt{PExtract}(\mathsf{pp}, \mathsf{pk}_{i^\star}), \texttt{PExtract}(\mathsf{pp}, \mathsf{pk}_{j^\star})$ corresponding to the challenge public keys. In order to maximize the probability that D is spanning *and* that we can at the same time successfully extract the secret keys $\mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$ from the reduction with large probability, we choose $|D^\star| = N/2$. Now, the strategy for the simulated adversary is to first choose the "real" challenge pair $\{i^\star, j^\star\}$, and then rewind the reduction for all possible choices of $\{i^\star, j^\star\} \notin D$. Thereby, the simulated adversary will obtain the secret keys $\mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$, which it can use to check validity and consistency with the other key pairs in the system and finally to compute the shared key $K^\star$ of the real challenge pair. With validity and consistency of the involved key pairs, and the fact that D is spanning, Lemma 4.5 and Corollary 4.6 guarantee that $K^\star$ computed by $\mathcal{B}$ is equal to the key computed by the hypothetical adversary. Thus, $\mathcal{B}$ perfectly simulates the hypothetical adversary unless D is not spanning, or the reduction does abort on all "fake" runs on which the reduction could have extracted $\mathsf{sk}_{i^\star}$ or $\mathsf{sk}_{j^\star}$ (but not on the real run).

More formally, we can describe the following simulated adversary $\mathcal{B}$ attempting to break $\mathcal{N} = (G, U, V)$. $\mathcal{B}$ runs the reduction $\Lambda = (\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4)$ and internally simulates $\mathcal{A}$ as follows: Let c be the input of $\mathcal{B}$, where $(c, w) \leftarrow G(1^\lambda)$. The adversary $\mathcal{B}$ starts with tossing a coin χ with $\Pr[\chi = 1] = \varepsilon_{\mathcal{A}}$. If $\chi = 0$, $\mathcal{B}$ sets $\mathsf{out}_{\mathcal{A}} = \perp$.

1. The adversary $\mathcal{B}$ runs $(\mathcal{PP}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N, st_{\Lambda_1}) \leftarrow \Lambda_1(c, r_\lambda)$.

2. The adversary $\mathcal{B}$ samples $D^\star \subset [N]$ of size $|D^\star| = N/2$ and $C^\star \subseteq D^\star$ of size $|C^\star| = 2$ uniformly at random and sets $D = D^\star \backslash C^\star$.

3. $\mathcal{B}$ runs $((\mathsf{sk}_i)_{i \in D}, st_{\Lambda_2}) \leftarrow \Lambda_2(D, st_{\Lambda_1})$. Next, $\mathcal{B}$ checks whether it holds $\texttt{Ver}(\mathsf{pk}_i, \mathsf{sk}_i) = 1$ and $\texttt{SharedKey}(\mathsf{pp}, \mathsf{pk}_j, \mathsf{sk}_i) = \texttt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_j)$ for all $i, j \in D$ with $i \neq j$. If any of the checks fail, it sets $\mathsf{out}_{\mathcal{A}} = \perp$.

4. The adversary initializes $R_{\mathsf{valid}} = \emptyset$ to save the set of valid runs for which the hypothetical adversary would not have aborted, and $K_{\mathsf{valid}} = \emptyset$ to save the keys extracted by the reduction via rewinding. For each $C \subset [N] \backslash D$ of size $|C| = 2$ the adversary runs the reduction $\Lambda_3(C, st_{\Lambda_2})$. Let $I = [n] \backslash (D \cup C)$. Let $((\mathsf{sk}_i)_{i \in I}, st^C_{\Lambda_3})$ denote the output of the reduction Λ for the respective execution. Whenever $\texttt{Ver}(\mathsf{pk}_i, \mathsf{sk}_i) = 1$ and $\texttt{SharedKey}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_j) = \texttt{SharedKey}(\mathsf{pp}, \mathsf{pk}_j, \mathsf{sk}_i)$ for all $i, j \in D \cup C$ with $i \neq j$, the adversary sets $R_{\mathsf{valid}} = R_{\mathsf{valid}} \cup \{C\}$ and $K_{\mathsf{valid}} = K_{\mathsf{valid}} \cup \{(i, \mathsf{sk}_i)\}_{i \in [n] \backslash (C \cup D)}$.

5. If $C^\star \notin R_{\mathsf{valid}}$ or there does not exist a pair of secret keys $\mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$ such that $(i^\star, \mathsf{sk}_{i^\star}), (j^\star, \mathsf{sk}_{j^\star}) \in K_{\mathsf{valid}}$, $\mathcal{B}$ sets $\mathsf{out}_{\mathcal{A}} = \perp$. Else, let $C^\star := \{i^\star, j^\star\}$. Then, $\mathcal{B}$ computes $K^\star = \texttt{SharedKey}(\mathsf{pk}_{j^\star}, \mathsf{sk}_{i^\star})$.

6. Finally, if $\mathsf{out}_{\mathcal{A}} = \perp$, the adversary $\mathcal{B}$ outputs $s \xleftarrow{\$} \Lambda_4(\perp, st^{C^\star}_{\Lambda_3})$. Otherwise, it outputs $s \xleftarrow{\$} \Lambda_4(K^\star, st^{C^\star}_{\Lambda_3})$.

Efficiency of $\mathcal{B}$. In the fourth step Λ_3 has to be executed $\binom{N/2}{2}$ times. For each $i \in [N]$, the validity check has to be performed once. Further, the reduction has to check validity and compute a shared key. The running time $\mathcal{B}$ can thus be (loosely) upper bounded by

$$T(\mathcal{B}) \leq N^2 \cdot T(\Lambda) + N^2 \cdot (N \cdot T(\texttt{Ver}) + N^2 \cdot T(\texttt{SharedKey})) \leq \mathrm{poly}(\lambda).$$

We stress that $\mathcal{B}$ itself does not have to satisfy any particular efficiency bound (e.g., to meet a tightness criterion). In fact, the theorem statement only asks for a polynomial-time $\mathcal{B}$.

Success probability of $\mathcal{B}$. Let $C^\star = \{i^\star, j^\star\}$ as before. Consider the following three events:

$$\begin{aligned}
\mathsf{spanning}: &\quad \text{for } \mathbf{x}_i \leftarrow \texttt{PExtract}(\mathsf{pp}, \mathsf{pk}_i) \text{ it holds } \mathbf{x}_{i^\star}, \mathbf{x}_{j^\star} \text{ are in the span of } \{\mathbf{x}_i\}_{i \in D} \\
\mathsf{check\text{-}valid}: &\quad C^\star \in R_{\mathsf{valid}} \\
\mathsf{challenge\text{-}valid}: &\quad \exists \mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}: (i^\star, \mathsf{sk}_{i^\star}), (j^\star, \mathsf{sk}_{j^\star}) \in K_{\mathsf{valid}}
\end{aligned}$$

Hence, spanning occurs when the vectors corresponding to set D span the space containing the vectors $\mathbf{x}_{i^\star}, \mathbf{x}_{j^\star}$ corresponding to the challenge. The event check-valid occurs if the reduction would not have aborted in the "real" run. challenge-valid occurs if the reduction provided secret keys for both challenge indices $i^\star$ and $j^\star$ in some rewinding attempt such that these key pairs are valid and consistent with all key pairs in D, thereby allowing to compute the (unique) shared key.

These events are naturally defined in runs with $\mathcal{B}$. However, in the following, we will also require these events for case distinctions in runs of the reduction $\Lambda^{\mathcal{A}}$ with the hypothetical adversary $\mathcal{A}$. We interpret these events then as a function of the reduction Λ's random coins and the set $D^\star$.

Note that if $\chi = 0$, then $\mathcal{B}$ simulates the adversary $\mathcal{A}$ perfectly, as both will abort. For case $\chi = 1$, we first consider the case $\mathsf{spanning} \wedge (\neg\mathsf{check\text{-}valid} \vee (\mathsf{check\text{-}valid} \wedge \mathsf{challenge\text{-}valid}))$. If $\neg\mathsf{check\text{-}valid}$, then $\mathcal{B}$ returns $\perp$ in step 5. The hypothetical adversary would have returned $\perp$ as well, since this means the reduction did not return valid and consistent key pairs on the real run. If $\mathsf{check\text{-}valid} \wedge \mathsf{challenge\text{-}valid}$, then the reduction provided N secret keys $(\mathsf{sk}_i)_{i\in D}, (\mathsf{sk}_i)_{i\in C}, \mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$ witnessing validity of $(\mathsf{pk}_i)_{i\in[n]}$. Further, we are ensured that the secret keys $\mathsf{sk}_{i^\star}, \mathsf{sk}_{j^\star}$ are consistent with all key pairs in D. Therefore, Lemma 4.5 and Corollary 4.6 yield that the key shared among $\mathsf{pk}_{i^\star}$ and $\mathsf{pk}_{j^\star}$ does not depend on the choice of secret key among all the ones that witness the validity and consistency properties with the keys pairs in D. Hence, in case $\mathsf{check\text{-}valid} \wedge \mathsf{challenge\text{-}valid}$ the simulated adversary always outputs the correct shared key $K^\star$, as does the hypothetical adversary.

We summarize all other possible cases in the event

$$\mathsf{bad} = \mathsf{bad}_1 \vee \mathsf{bad}_2,$$

where

$$\mathsf{bad}_1 = \neg\mathsf{spanning}$$

and

$$\mathsf{bad}_2 = \neg(\neg\mathsf{check\text{-}valid} \vee (\mathsf{check\text{-}valid} \wedge \mathsf{challenge\text{-}valid})) = \mathsf{check\text{-}valid} \wedge \neg\mathsf{challenge\text{-}valid},$$

which are well-defined, since Λ_3 is deterministic.

We start by bounding the probability of bad_1. Recall that the vectors $\mathbf{x}_i$ are of dimension $d \leq N/2 - 2$. Therefore, for any possible set $\{\mathbf{x}_i\}_{i\in D^\star}$ there exists a subset $S \subset D^\star$ such that $|S| \leq d \leq N/2 - 2$ and $\{\mathbf{x}_i\}_{i\in S}$ spans $\{\mathbf{x}_i\}_{i\in D^\star}$. Now for a random challenge set $C^\star \subset D^\star$ of size 2, the probability that $C^\star \subset D^\star \backslash S$ can be computed as

$$\Pr[C^\star \cap S = \emptyset] = \frac{|D^\star| - |S|}{|D^\star|} \cdot \frac{|D^\star| - |S| - 1}{|D^\star| - 1} \geq \frac{(N/2 - d)\cdot(N/2 - d - 1)}{N/2 \cdot (N/2 - 1)} \geq \frac{N^2/4 - 2\cdot d(N/2) - N/2}{N^2/4}$$
$$= 1 - \frac{dN + N/2}{N^2/4} = 1 - \frac{4d+2}{N}.$$

We can thus bound the event bad_1 as

$$\Pr[\mathsf{bad}_1] = 1 - \Pr[\mathsf{spanning}] \leq \frac{4d+2}{N}.$$

It is left to bound the probability of bad_2. First note that we have

$$\Pr[\neg\mathsf{challenge\text{-}valid}] \leq \Pr[\mathsf{sk\text{-}bad}],$$

where $\mathsf{sk\text{-}bad}$ is the event that there exists an index $k^\star \in [N]\backslash D$ such that $(k^\star, *) \notin K_{\mathsf{valid}}$. Per definition of events, if $\mathsf{sk\text{-}bad}$ and $k^\star \notin C^\star$, then $\mathsf{check\text{-}valid}$ cannot occur. We thus have

$$\Pr[\mathsf{bad}_2] \leq \Pr[\mathsf{check\text{-}valid} \mid \mathsf{sk\text{-}bad}] \cdot \underbrace{\Pr[\mathsf{sk\text{-}bad}]}_{\leq 1}$$
$$\leq \Pr[k^\star \in C^\star] \leq \frac{2}{N/2} = \frac{4}{N},$$

where the last inequality follows because $C^\star$ is chosen uniformly at random and independent of the view of the reduction and via a union bound over $k^\star = i^\star$ or $k^\star = j^\star$.

Altogether, we obtain

$$\Pr[\mathsf{bad}] \leq \frac{4d+2}{N} + \frac{4}{N} = \frac{4d+6}{N}.$$

For any adversary on the underlying NICA let $\mathsf{S}(\cdot)$ denote the event that an adversary successfully breaks the NICA. Then, the above yields:

$$\Pr[\mathsf{S}(\Lambda^{\mathcal{A}}) \mid \chi = 0] = \Pr[\mathsf{S}(\mathcal{B}) \mid \chi = 0] \tag{4}$$

and

$$|\Pr[\mathsf{S}(\Lambda^{\mathcal{A}}) \mid \chi = 1] - \Pr[\mathsf{S}(\mathcal{B}) \mid \chi = 1]| \leq \Pr[\mathsf{bad}] \leq \frac{4d+6}{N}. \tag{5}$$

Recall that U is the PPT machine that "trivially" breaks the NICA. Altogether, we obtain

$$\begin{aligned}
\varepsilon_\Lambda &= \left|\Pr[\mathsf{S}(\Lambda^{\mathcal{A}})] - \Pr[\mathsf{S}(U)]\right| \\
&= \left|\varepsilon_{\mathcal{A}} \cdot \Pr[\mathsf{S}(\Lambda^{\mathcal{A}}) \mid \chi = 1] + (1 - \varepsilon_{\mathcal{A}}) \cdot \Pr[\mathsf{S}(\Lambda^{\mathcal{A}}) \mid \chi = 0] - \Pr[\mathsf{S}(U)]\right| \\
&\overset{\text{Eq. 4}}{=} \left|\varepsilon_{\mathcal{A}} \cdot \Pr[\mathsf{S}(\Lambda^{\mathcal{A}}) \mid \chi = 1] + (1 - \varepsilon_{\mathcal{A}}) \cdot \Pr[\mathsf{S}(\mathcal{B}) \mid \chi = 0] - \Pr[\mathsf{S}(U)]\right| \\
&\overset{\text{Eq. 5}}{\leq} |\varepsilon_{\mathcal{A}} \cdot (\Pr[\mathsf{S}(\mathcal{B}) \mid \chi = 1] + \Pr[\mathsf{bad}]) + (1 - \varepsilon_{\mathcal{A}}) \cdot \Pr[\mathsf{S}(\mathcal{B}) \mid \chi = 0] - \Pr[\mathsf{S}(U)]| \\
&\leq \varepsilon_{\mathcal{A}} \cdot \Pr[\mathsf{bad}] + |\Pr[\mathsf{S}(\mathcal{B})] - \Pr[\mathsf{S}(U)]| \\
&\overset{\text{Eq. 5}}{\leq} \varepsilon_{\mathcal{A}} \cdot \frac{4d+6}{N} + \mathsf{Adv}^{\mathsf{nica}}_{\mathcal{N},\mathcal{B}}.
\end{aligned}$$

□

Interpretation. Theorem 4.7 says that if any reduction is successfully breaking the underlying NICA $\mathcal{N}$ with probability noticeably larger than $(4d + 6)/N$, the reduction can be turned into a standalone $\mathcal{N}$ solver, without help of an external adversary. More precisely, assuming $\mathcal{N}$ is hard we obtain

$$\varepsilon_\Lambda \leq \frac{4d+6}{N} \cdot \varepsilon_{\mathcal{A}} + \text{negl}$$

for a negligible function negl. This implies a security loss of at least $N/(4d + 6)$.

We can thus conclude that any inner-product NIKE that satisfies 2-step-adaptive security has to either have a significant loss, or ip-dimension proportional to the number of users N. In particular, this gives strong evidence that a fully-adaptive NIKEs with tight security only exist for an a priori fixed number of users, but not for a dynamic setting where users continuously join or leave. Altogether, using the relations between security notions depicted in Figure 5, we obtain the following informal corollary:

Corollary 4.8 *Any simple reduction from a non-interactive complexity assumption $\mathcal{N}$ to the X-security of a d-dimensional ip-NIKE has to lose a factor in the order of Y, where N is the number of public keys, $\mathcal{N}$ is assumed to be hard and $(X, Y) \in \{(2\text{-step-adaptive}, \Omega(N/d)), (\text{adaptive}, \Omega(N/d)), (\nu\text{-semi-adaptive}, \Omega(\nu^2/(N \cdot d))\}$.*

Our lower bound thus yields that the core NIKE construction with secret keys of size $O(\nu)$ has an inherent security loss of at least $\Omega(N/\nu)$. Recall that the security loss we showed in Theorem 3.2 is $O((N/\nu)^2 \log \nu)$. Thus, for $\nu = N$ our upper and lower bound on the security loss are essentially matching (except for the logarithmic factor). It is an interesting open question to close the gap between the two bounds for $\nu < N$.

Note that setting $\nu = d$, the lower bound on ν-semi-adaptive security simplifies to $\Omega(\nu/N) \leq \Omega(1)$ and thus does not allow any meaningful statement on tightness. Hence, one could still hope for semi-adaptive yet tight NIKE constructions with smaller key sizes. And indeed, in the upcoming section we show that one can obtain a trade-off between public-key size and adaptivity without compromising on tightness. We remark that this upcoming NIKE construction escapes our lower bound by not quite fitting in the form of an inner-product NIKE.

Finally, we want to point out that having high ip-dimension alone does not imply that the public and secret keys have to be large, since the dimension-d vectors could be stored in a compressed way. But for all examples of ip-NIKEs from the literature, the public and secret keys will indeed scale with the ip-dimension. In these cases our lower bound indeed gives a trade-off between the tightness of the reduction and the size of the keys.

5 Semi-adaptively secure NIKE

Our NIKE construction $\mathtt{NIKE}_{\mathsf{ip}}$ from Section 3 has an additional helpful security property: It not only guarantees that for the first ν users (where ν is the key size parameter) the shared keys all look uniformly random to the adversary, it actually allows the construction to choose the shared keys. Concretely, when the adversary makes a test query, the reduction could use an (external) source to obtain an element $x \in \mathbb{Z}_q$ that looks uniformly random to the adversary and then answer the test query with $[x]_T$. And, even if the limit of ν involved users is exceeded, the reduction can answer all queries consistently (only the shared keys are then all correlated with each other).

To emphasize that our "shared keys" do not provide any secrecy, we will call them "shared tags" instead and the mechanism a non-interactive tag exchange (NITE) scheme. The following definition formalizes this security property. A $\mathtt{NITE}$ can be operated in two modes. The normal mode works like a regular NIKE: $\mathtt{Setup}$ generates a public parameters, $\mathtt{KeyGen}$ can be used to generate key pairs, and $\mathtt{SharedTag}$ computes a tag from a public key and a secret key of two different users. The other mode is the programming mode, where the stateful algorithms $\mathtt{Setup}'$, $\mathtt{RegH}$, $\mathtt{Test}$, $\mathtt{RevH}$, $\mathtt{Extr}$ are used. Again, $\mathtt{Setup}'$ generates the public parameters, $\mathtt{RegH}$ registers a user and returns its public key, and $\mathtt{Extract}$ and $\mathtt{RevH}$ can be used to open secret keys and shared keys, respectively. $\mathtt{Test}$ can be used to program the shared tag between two users to a specific value, as long as less than ν users have been used in a $\mathtt{Extract}$, $\mathtt{RevH}$, and $\mathtt{Test}$ calls so far. For this programming procedure, we equip the tag space $\mathcal{T}$ with a tag description space $\mathcal{H}$ and an efficiently computable bijection $f : \mathcal{H} \to \mathcal{T}$. To program a tag to $f(x)$, you have to call $\mathtt{Test}$ with x and the identities of the two users. In our construction, $\mathcal{T}$ will be $\mathbb{G}_T$, $\mathcal{H}$ will be $\mathbb{Z}_q$, and f will map x to $[x]_T$. This is necessary, because the programming procedure of our $\mathtt{NITE}$ needs the value x over $\mathbb{Z}_q$ to program a shared key to $[x]_T$.

Security now guarantees, that an adversary can not distinguish the normal mode and the programming mode, as long as in all calls of $\mathtt{Test}$ use a value $x \in \mathcal{H}$ that is an independent uniformly random value in the adversaries point of view.

Definition 5.1 (NITE). A *NITE* scheme with identity space $\mathcal{IDS}$ and tag space $(f, \mathcal{H}, \mathcal{T})$ consists of seven PPT algorithms $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedTag}, \mathtt{Setup}', \mathtt{RegH}, \mathtt{Test}, \mathtt{RevH}, \mathtt{Extr})$, where

- $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedTag})$ form syntactically a NIKE with key space $\mathcal{T}$.
- $\mathtt{Setup}'$ is an alternative way to generate the public parameters. It takes the unary encoded security parameter 1^λ and outputs public parameters pp and a state *st*.
- $\mathtt{RegH}$ takes the state *st* and an identity id and outputs public key pk and an updated state *st*.
- $f : \mathcal{H} \to \mathcal{T}$ is a efficiently computable bijection.
- $\mathtt{Test}$ takes the state *st* along with two challenge identities $\mathsf{id}_1^\star$, $\mathsf{id}_2^\star$, and tag description $h_\ell^\star \in \mathcal{H}$. It outputs an updated state *st*.
- $\mathtt{RevH}$ takes the state *st* along with two challenge identities id_1, id_2 and outputs a shared key T for the users id_1 and id_2 and an updated state *st*.
- $\mathtt{Extr}$ takes the state *st* and an identity id and outputs a secret key sk for id and an updated state *st*.

Definition 5.2 (Correctness). A NITE $\mathtt{NITE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedTag}, \mathtt{Setup}', \mathtt{RegH}, \mathtt{Test}, \mathtt{RevH}, \mathtt{Extr})$ is correct, iff $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedTag})$ is a correct NIKE.

In the security game, there is no need to distinguish between an oracle that reveals honest keys and a challenge oracle. Thus we provide only one oracle $\mathcal{O}_{\mathsf{test}}$ that will program the shared keys until more than ν users are involved and returns non-programmed shared keys afterwards for simplicity.

Definition 5.3 (ν-programmable security). We say that a NITE $\mathtt{NITE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedTag}, \mathtt{Setup}', \mathtt{RegH}, \mathtt{Test}, \mathtt{RevH}, \mathtt{Extr})$ is ν-programmable-secure, if for all PPT adversaries $\mathcal{A}$

$$\mathsf{Adv}^{\nu\text{-programmable}}_{\mathcal{A},\mathtt{NITE}}(\lambda) := |\Pr[\mathsf{G}^{\mathsf{real}}_{\mathcal{A},\mathtt{NITE}} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathsf{programmed}}_{\mathcal{A},\mathtt{NITE}} \Rightarrow 1]|$$

is negligible. The games $\mathsf{G}^{\mathsf{real}}_{\mathcal{A},\mathtt{NITE}}$ and $\mathsf{G}^{\mathsf{programmed}}_{\mathcal{A},\mathtt{NITE}}$ are defined in Figure 10.

We can use the programmable tags to build an ordinary NIKE. Therefore we need another component which we call tagged non-interactive key exchange (TNIKE), which is essentially the same as a NIKE except that the $\mathtt{SharedKey}$ algorithm gets additionally a tag as input. Correctness only holds when both parties use the same tag (that has to be independent of public parameters and keys involved), and security holds only for one specific secret tag.

```
$\mathsf{G}^{\mathsf{real}}_{\mathcal{A},\mathtt{NITE}}$:
    $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset;\ Q_{\mathsf{inv}} := \emptyset;\ Q_{\mathsf{reg}} := \emptyset$
    $\mathsf{pks} : \mathcal{IDS} \dashrightarrow \mathcal{PK}$
    $\mathsf{sks} : \mathcal{IDS} \dashrightarrow \mathcal{SK}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{id} \in Q_{\mathsf{reg}}$ then return $\perp$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id})$
    $Q_{\mathsf{reg}} := Q_{\mathsf{reg}} \cup \{\mathsf{id}\};\ \mathsf{pks}(\mathsf{id}) := \mathsf{pk};\ \mathsf{sks}(\mathsf{id}) := \mathsf{sk}$
    return $\mathsf{pk}$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{id} \notin Q_{\mathsf{reg}}$ then return $\perp$
    $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$
    return $\mathsf{sks}(\mathsf{id})$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{id}_1^\star, \mathsf{id}_2^\star \in Q_{\mathsf{reg}}$ then
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\}$
        $Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1^\star);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2^\star)$
        $T^\star \leftarrow \mathtt{SharedTag}(\mathsf{pp}, \mathsf{id}_1^\star, \mathsf{pk}_1, \mathsf{id}_2^\star, \mathsf{sk}_2)$
        return $T^\star$
    return $\perp$
```

```
$\mathsf{G}^{\mathsf{programmed}}_{\mathcal{A},\mathtt{NITE}}$:
    $(\mathsf{pp}, st) \leftarrow \mathtt{Setup}'(1^\lambda)$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset;\ Q_{\mathsf{inv}} := \emptyset;\ Q_{\mathsf{reg}} := \emptyset$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{id} \in Q_{\mathsf{reg}}$ then return $\perp$
    $Q_{\mathsf{reg}} := Q_{\mathsf{reg}} \cup \{\mathsf{id}\}$
    $(\mathsf{pk}, st) \leftarrow \mathtt{RegH}(st, \mathsf{id})$
    return $\mathsf{pk}$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{id} \notin Q_{\mathsf{reg}}$ then return $\perp$
    $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$
    $(\mathsf{sk}, st) \leftarrow \mathtt{Extr}(st, \mathsf{id})$
    return $\mathsf{sk}$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{id}_1^\star, \mathsf{id}_2^\star \in Q_{\mathsf{reg}}$ then
        $Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$
        if $|Q_{\mathsf{inv}}| \le \nu \wedge \mathsf{id}_1^\star, \mathsf{id}_2^\star \notin Q_{\mathsf{extr}} \wedge \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\} \notin Q_{\mathsf{test}}$
        then
            $h_\ell^\star \xleftarrow{\$} \mathcal{H};\ T^\star := f(h_\ell^\star)$
            $st \leftarrow \mathtt{Test}(st, h_\ell^\star, \mathsf{id}_1^\star, \mathsf{id}_2^\star)$
        else
            $(T^\star, st) \leftarrow \mathtt{RevH}(st, i, j)$
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\}$
        return $T^\star$
    return $\perp$
```

Figure 10: Games for defining ν-programmable security of a NITE scheme NITE. The partial maps pks and sks are initially totally undefined.

```
$\mathsf{Exp}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{rev}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset;\ Q_{\mathsf{inv}} := \emptyset$
    $\mathsf{pks} : \mathcal{IDS} \dashrightarrow \mathcal{PK}$
    $\mathsf{sks} : \mathcal{IDS} \dashrightarrow \mathcal{SK}$
    $b \xleftarrow{\$} \{0,1\}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{revH}}(\cdot,\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$
    if $Q_{\mathsf{rev}} \cap Q_{\mathsf{test}} = \emptyset \wedge \nexists A \in Q_{\mathsf{test}} : A \cap Q_{\mathsf{extr}} = \emptyset$ then
        return $b \overset{?}{=} b^\star$
    else
        return $0$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}) \neq \bot$ then return $\bot$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id})$
    $\mathsf{pks}(\mathsf{id}) := \mathsf{pk};\ \mathsf{sks}(\mathsf{id}) := \mathsf{sk}$
    return $\mathsf{pk}$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$
    return $\mathsf{sks}(\mathsf{id})$

$\mathcal{O}_{\mathsf{revH}}(\mathsf{id}_1 \in \mathcal{IDS}, \mathsf{id}_2 \in \mathcal{IDS}, T \in \mathcal{T})$:
    if $\mathsf{pks}(\mathsf{id}_1) \neq \bot \wedge \mathsf{sks}(\mathsf{id}_2) \neq \bot$ then
        $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{id}_1, \mathsf{id}_2\}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1, \mathsf{id}_2\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2)$
        return $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2, T)$
    return $\bot$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1^\star) \neq \bot \wedge \mathsf{sks}(\mathsf{id}_2^\star) \neq \bot \wedge \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\} \notin Q_{\mathsf{test}}$ then
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$
        if $|Q_{\mathsf{inv}}| \leq \nu$ then
            $(i, j) := (\arg\max_{i \in \{1,2\}} \mathsf{id}_i^\star, \arg\min_{i \in \{1,2\}} \mathsf{id}_i^\star)$
            $\mathsf{pk}_i := \mathsf{pks}(\mathsf{id}_i^\star);\ \mathsf{sk}_j := \mathsf{sks}(\mathsf{id}_j^\star)$
            $L \leftarrow \mathtt{SecretTags}(\mathsf{sk}_i)$
            $h_\ell^\star := L[|Q_{\mathsf{test}}|];\ T^\star := f(h_\ell^\star)$
            $K_0^\star \leftarrow \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_i^\star, \mathsf{pk}_i, \mathsf{id}_j^\star, \mathsf{sk}_j, T^\star)$
            $K_1^\star \xleftarrow{\$} \mathcal{K}$
            return $K_b^\star$
    return $\bot$
```

Figure 11: Experiment for ν-semi-adaptive security of a TNIKE scheme $\mathtt{TNIKE}$ with shared key space $\mathcal{K}$. $\mathcal{PK}$ denotes the public key space and $\mathcal{SK}$ denotes the secret key space. The partial maps pks and sks are initially totally undefined. Remember that $L[|Q_{\mathsf{test}}|]$ stands for the $|Q_{\mathsf{test}}|$-th element in the list L.

Definition 5.4 (TNIKE). A *TNIKE* scheme with tag space $(f, \mathcal{H}, \mathcal{T})$ and key space $\mathcal{K}$ consists of four PPT algorithms $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey}, \mathtt{SecretTags})$, where

- $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey})$ is a NIKE with key space $\mathcal{K}$, except that $\mathtt{SharedKey}$ additionally takes a tag $T \in \mathcal{T}$ as input and
- $f : \mathcal{H} \to \mathcal{T}$ is a efficiently computable bijection.
- $\mathtt{SecretTags}$ is a deterministic algorithm that takes a secret key sk and outputs a list of special "secret" tag descriptions $L \in \mathcal{H}^*$.

Definition 5.5 (Correctness). We say that a TNIKE $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey}, \mathtt{SecretTags})$ for identity space $\mathcal{IDS}$ is *statistically correct*, if the correctness error

$$\sup_{\mathsf{id}_1, \mathsf{id}_2 \in \mathcal{IDS}, T \in \mathcal{T}} \Pr[\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2, T) \neq \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_2, \mathsf{pk}_2, \mathsf{id}_1, \mathsf{sk}_1, T) \mid \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id}_2)]$$

is negligible in λ.

Definition 5.6 (ν-semi-adaptive security). We say that a TNIKE $\mathtt{TNIKE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey}, \mathtt{SecretTags})$ is ν-semi-adaptively secure, if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$

$$\mathsf{Adv}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda) := 2\Pr[\mathsf{Exp}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda) \Rightarrow 1] - 1$$

is negligible and $\mathtt{SecretTags}(\mathsf{sk}_i)$ generates a list of at least ν tag descriptions where each of them has only negligible statistical distance $\varepsilon_{\mathsf{hidden}}(\lambda)$ to a uniformly random value that is independent of the public parameters pp, the corresponding public key pk_i and the other tag descriptions in the list. We call $\varepsilon_{\mathsf{hidden}}(\lambda)$ the tag hiding error. The game $\mathsf{Exp}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda)$ is defined in Figure 11.

```
$\mathtt{Setup}(1^\lambda)$:
    $\mathsf{pp}_{\mathsf{NITE}} \leftarrow \mathtt{Setup}_{\mathsf{NITE}}(1^\lambda)$
    $\mathsf{pp}_{\mathsf{TNIKE}} \leftarrow \mathtt{Setup}_{\mathsf{TNIKE}}(1^\lambda)$
    $\textbf{return } \mathsf{pp} := (\mathsf{pp}_{\mathsf{NITE}}, \mathsf{pp}_{\mathsf{TNIKE}})$

$\mathtt{KeyGen}(\mathsf{pp}, \mathsf{id}_{\mathsf{id}})$:
    $\textbf{parse } \mathsf{pp} =: (\mathsf{pp}_{\mathsf{NITE}}, \mathsf{pp}_{\mathsf{TNIKE}})$
    $(\mathsf{pk}_{\mathsf{NITE},}, \mathsf{sk}_{\mathsf{NITE},}) \leftarrow \mathtt{KeyGen}_{\mathsf{NITE}}(\mathsf{pp}_{\mathsf{NITE}}, \mathsf{id})$
    $(\mathsf{pk}_{\mathsf{TNIKE}}, \mathsf{sk}_{\mathsf{TNIKE}}) \leftarrow \mathtt{KeyGen}_{\mathsf{TNIKE}}(\mathsf{pp}_{\mathsf{TNIKE}}, \mathsf{id})$
    $\mathsf{pk} := (\mathsf{pk}_{\mathsf{NITE}}, \mathsf{pk}_{\mathsf{TNIKE}});\ \mathsf{sk} := (\mathsf{sk}_{\mathsf{NITE}}, \mathsf{sk}_{\mathsf{TNIKE}})$
    $\textbf{return } (\mathsf{pk}, \mathsf{sk})$
```

```
$\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2)$:
    $\textbf{parse } \mathsf{pp} =: (\mathsf{pp}_{\mathsf{NITE}}, \mathsf{pp}_{\mathsf{TNIKE}})$
    $\textbf{parse } \mathsf{pk}_1 =: (\mathsf{pk}_{\mathsf{NITE},1}, \mathsf{pk}_{\mathsf{TNIKE},1})$
    $\textbf{parse } \mathsf{sk}_2 =: (\mathsf{sk}_{\mathsf{NITE},2}, \mathsf{sk}_{\mathsf{TNIKE},2})$
    $T \leftarrow \mathtt{SharedTag}(\mathsf{pp}_{\mathsf{NITE}}, \mathsf{id}_1, \mathsf{pk}_{\mathsf{NITE},1}, \mathsf{id}_2, \mathsf{sk}_{\mathsf{NITE},2})$
    $\textbf{return } K \leftarrow \mathtt{SharedKey}_{\mathsf{TNIKE}}(\mathsf{pp}_{\mathsf{TNIKE}}, \mathsf{id}_1, \mathsf{pk}_{\mathsf{TNIKE},1}, \mathsf{id}_2, \mathsf{sk}_{\mathsf{TNIKE},2}, T)$
```

Figure 12: The transformation of a NITE $\mathtt{NITE} = (\mathtt{Setup}_{\mathsf{NITE}}, \mathtt{KeyGen}_{\mathsf{NITE}}, \mathtt{SharedTag}, \mathtt{Setup}', \mathtt{RegH}, \mathtt{Test}, \mathtt{RevH}, \mathtt{Extr})$ and a TNIKE $\mathtt{TNIKE} = (\mathtt{Setup}_{\mathsf{TNIKE}}, \mathtt{KeyGen}_{\mathsf{TNIKE}}, \mathtt{SharedKey}_{\mathsf{TNIKE}}, \mathtt{SecretTags})$ to a NIKE $\mathtt{NIKE}_{\mathsf{sa}} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey})$.

Note that in the security experiment for ν-semi-adaptive some fixed, public ordering of the identities is used to determine which of the challenge identities contributes the public key and which contributes the secret key to the computation of the shared key. If the adversary could choose this, our security proof would not work.

5.1 The transformation

In Figure 12 we present a generic transformation from a ν-programmable secure NITE with tag space $(f, \mathcal{H}, \mathcal{T})$ and a ν-semi-adaptively secure TNIKE with the same tag space $(f, \mathcal{H}, \mathcal{T})$ to a ν-semi-adaptively secure NIKE $\mathtt{NIKE}_{\mathsf{sa}}$. Essentially, $\mathtt{NIKE}_{\mathsf{sa}}$ uses the NITE scheme to generate a tag for each pair of users and uses this tag for the TNIKE scheme to generate the secret keys.

Theorem 5.7 (Correctness). *When* $\mathtt{NITE}$ *is correct (with correctness error* $\varepsilon_{\mathsf{NITE}}(\lambda)$*) and* $\mathtt{TNIKE}$ *is correct (with correctness error* $\varepsilon_{\mathsf{TNIKE}}(\lambda)$*), then* $\mathtt{NIKE}_{\mathsf{sa}}$ *is a correct NIKE (with correctness error* $\varepsilon_{\mathsf{NITE}}(\lambda) + \varepsilon_{\mathsf{TNIKE}}(\lambda)$*).*

Proof. Suppose no correctness error occurs in the NITE. Then $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2)$ and $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_2, \mathsf{pk}_2, \mathsf{id}_1, \mathsf{sk}_1)$ both compute the same tag T and this tag is independent of $\mathsf{pp}_{\mathsf{TNIKE}}$, $\mathsf{pk}_{\mathsf{TNIKE},1}$, $\mathsf{sk}_{\mathsf{TNIKE},1}$, $\mathsf{pk}_{\mathsf{TNIKE},2}$, and $\mathsf{sk}_{\mathsf{TNIKE},2}$. With this in mind, the correctness of $\mathtt{NIKE}[\mathtt{NITE}, \mathtt{TNIKE}]$ follows directly from the correctness of $\mathtt{TNIKE}$. □

Theorem 5.8 (Security). *For every PPT adversary* $\mathcal{A}$ *against the* ν*-semi-adaptive security of* $\mathtt{NIKE}_{\mathsf{sa}}$*, there exists PPT adversaries* $\mathcal{B}$ *for the NITE* $\mathtt{NITE}$ *and* $\mathcal{C}$ *for the TNIKE* $\mathtt{TNIKE}$ *such that*

$$\mathsf{Adv}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{NIKE}_{\mathsf{sa}}}(\lambda) \leq 2(\mathsf{Adv}^{\nu\text{-programmable}}_{\mathcal{B},\mathtt{NITE}}(\lambda) + \mathsf{Adv}^{\nu\text{-semi-adaptive}}_{\mathcal{C},\mathtt{TNIKE}}(\lambda) + N_{\mathsf{test}}\varepsilon_{\mathsf{TNIKE}}(\lambda)) + \varepsilon_{\mathsf{hidden}}(\lambda),$$

where $\varepsilon_{\mathsf{TNIKE}}(\lambda)$ *denotes the correctness error of* $\mathtt{TNIKE}$*,* $\varepsilon_{\mathsf{hidden}}(\lambda)$ *is the tag hiding error of* $\mathtt{TNIKE}$*, and* N_{test} *is the maximum number of* $\mathcal{A}$*'s* $\mathcal{O}_{\mathsf{test}}$ *queries. Furthermore,* $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ *and* $T(\mathcal{C}) \approx T(\mathcal{A}) + N\,\mathrm{poly}'(\lambda)$ *for a polynomials* $\mathrm{poly}, \mathrm{poly}'$ *independent of* $\mathcal{A}$*.*

Proof. The proof uses a hybrid argument with hybrids G_0–G_3. The hybrids are given in Figure 13.

Lemma 5.9 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$)*. For every PPT adversary* $\mathcal{A}$ *there exists an PPT adversary* $\mathcal{B}$ *such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq \mathsf{Adv}^{\nu\text{-programmable}}_{\mathcal{B},\mathtt{NITE}}(\lambda).$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$*.*

```
$\mathsf{G}_0$   $\mathsf{G}_1$   $\mathsf{G}_2$   $\mathsf{G}_3$

$\mathsf{Exp}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{NIKE}}(\lambda)$:
    $\mathsf{pp}_{\mathsf{NITE}} \leftarrow \mathtt{Setup}_{\mathsf{NITE}}(1^\lambda)$
    $(\mathsf{pp}_{\mathsf{NITE}}, st) \leftarrow \mathtt{Setup}'(1^\lambda)$
    $\mathsf{pp}_{\mathsf{TNIKE}} \leftarrow \mathtt{Setup}_{\mathsf{TNIKE}}(1^\lambda)$
    $\mathsf{pp} := (\mathsf{pp}_{\mathsf{NITE}}, \mathsf{pp}_{\mathsf{TNIKE}})$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{rev}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset;\ Q_{\mathsf{inv}} := \emptyset$
    $\mathsf{pks} : \mathcal{IDS} \dashrightarrow \mathcal{PK}$
    $\mathsf{sks} : \mathcal{IDS} \dashrightarrow \mathcal{SK}$
    $b \xleftarrow{\$} \{0,1\}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{revH}}(\cdot,\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$
    if $Q_{\mathsf{rev}} \cap Q_{\mathsf{test}} = \emptyset \wedge \nexists A \in Q_{\mathsf{test}} : A \cap Q_{\mathsf{extr}} = \emptyset$ then
        return $b \overset{?}{=} b^\star$
    else
        return $0$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}) \neq \bot$ then return $\bot$
    $(\mathsf{pk}_{\mathsf{NITE},}, \mathsf{sk}_{\mathsf{NITE},}) \leftarrow \mathtt{KeyGen}_{\mathsf{NITE}}(\mathsf{pp}_{\mathsf{NITE}}, \mathsf{id})$
    $(\mathsf{pk}_{\mathsf{NITE}}, st) \leftarrow \mathtt{RegH}(st, \mathsf{id});\ \mathsf{sk}_{\mathsf{NITE}} := \bot$
    $(\mathsf{pk}_{\mathsf{TNIKE}}, \mathsf{sk}_{\mathsf{TNIKE}}) \leftarrow \mathtt{KeyGen}_{\mathsf{TNIKE}}(\mathsf{pp}_{\mathsf{TNIKE}}, \mathsf{id})$
    $\mathsf{pks}(\mathsf{id}) := (\mathsf{pk}_{\mathsf{NITE}}, \mathsf{pk}_{\mathsf{TNIKE}})$
    $\mathsf{sks}(\mathsf{id}) := (\mathsf{sk}_{\mathsf{NITE}}, \mathsf{sk}_{\mathsf{TNIKE},i})$
    return $(\mathsf{pk}_{\mathsf{NITE}}, \mathsf{pk}_{\mathsf{TNIKE}})$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$
    if $\mathsf{id} \in Q_{\mathsf{reg}}$ then
        $(\_, \mathsf{sk}_{\mathsf{TNIKE}}) := \mathsf{sks}(\mathsf{id})$
        $(\mathsf{sk}_{\mathsf{NITE}}, st) \leftarrow \mathtt{Extr}(st, \mathsf{id})$
        $\mathsf{sks}(\mathsf{id}) := (\mathsf{sk}_{\mathsf{NITE}}, \mathsf{sk}_{\mathsf{TNIKE}})$
    return $\mathsf{sks}(\mathsf{id})$

$\mathcal{O}_{\mathsf{revH}}(\mathsf{id}_1 \in \mathcal{IDS}, \mathsf{id}_2 \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1) \neq \bot \wedge \mathsf{sks}(\mathsf{id}_2) \neq \bot$ then
        $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{id}_1, \mathsf{id}_2\}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1, \mathsf{id}_2\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1);\ \mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2)$
        $T \leftarrow \mathtt{SharedTag}(\mathsf{pp}_{\mathsf{NITE}}, \mathsf{id}_1, \mathsf{pk}_{\mathsf{NITE},1}, \mathsf{id}_2, \mathsf{sk}_{\mathsf{NITE},2})$
        if $|Q_{\mathsf{inv}}| \leq \nu \wedge \mathsf{id}_1, \mathsf{id}_2 \notin Q_{\mathsf{extr}} \wedge \{\{\mathsf{id}_1, \mathsf{id}_2\}\} \notin Q_{\mathsf{test}}$ then
            $h_\ell \xleftarrow{\$} \mathcal{H};\ T := f(h_\ell)$
            $st \leftarrow \mathtt{Test}(st, h^\star_\ell, \mathsf{id}_1, \mathsf{id}_2)$
        else
            $(T, st) \leftarrow \mathtt{RevH}(st, i, j)$
        return $\mathtt{SharedKey}_{\mathsf{TNIKE}}(\mathsf{pp}_{\mathsf{TNIKE}}, \mathsf{id}_1, \mathsf{pk}_{\mathsf{TNIKE},1}, \mathsf{id}_2, \mathsf{sk}_{\mathsf{TNIKE},2}, T)$
    return $\bot$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}^\star_1 \in \mathcal{IDS}, \mathsf{id}^\star_2 \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}^\star_1) \neq \bot \wedge \mathsf{sks}(\mathsf{id}^\star_2) \neq \bot \wedge \{\mathsf{id}^\star_1, \mathsf{id}^\star_2\} \notin Q_{\mathsf{test}}$ then
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}^\star_1, \mathsf{id}^\star_2\}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}^\star_1, \mathsf{id}^\star_2\}$
        if $|Q_{\mathsf{inv}}| \leq \nu$ then
            $(i, j) := (1, 2)$
            $(i, j) := (\arg\max_{i \in \{1,2\}} \mathsf{id}^\star_i, \arg\min_{i \in \{1,2\}} \mathsf{id}^\star_i)$
            $\mathsf{pk}_i := \mathsf{pks}(\mathsf{id}^\star_i);\ \mathsf{sk}_j := \mathsf{sks}(\mathsf{id}^\star_j)$
            parse $\mathsf{pk}_i =: (\mathsf{pk}_{\mathsf{NITE},i}, \mathsf{pk}_{\mathsf{TNIKE},i})$
            parse $\mathsf{sk}_j =: (\mathsf{sk}_{\mathsf{NITE},j}, \mathsf{sk}_{\mathsf{TNIKE},j})$
            $T^\star \leftarrow \mathtt{SharedTag}(\mathsf{pp}_{\mathsf{NITE}}, \mathsf{id}^\star_1, \mathsf{pk}_{\mathsf{NITE},1}, \mathsf{id}^\star_2, \mathsf{sk}_{\mathsf{NITE},2})$
            $h^\star_\ell \xleftarrow{\$} \mathcal{H}$
            $L \leftarrow \mathtt{SecretTags}(\mathsf{sk}_i)$
            $h^\star_\ell := L[|Q_{\mathsf{test}}|]$
            $T^\star := f(h^\star_\ell)$
            $st \leftarrow \mathtt{Test}(st, h^\star_\ell, \mathsf{id}^\star_i, \mathsf{id}^\star_j)$
            $K^\star_0 \leftarrow \mathtt{SharedKey}_{\mathsf{TNIKE}}(\mathsf{pp}_{\mathsf{TNIKE}}, \mathsf{id}^\star_i, \mathsf{pk}_{\mathsf{TNIKE},i}, \mathsf{id}^\star_j, \mathsf{sk}_{\mathsf{TNIKE},j}, T^\star)$
            $K^\star_1 \xleftarrow{\$} \mathcal{K}$
            return $K^\star_b$
    return $\bot$
```

Figure 13: Hybrids for the security proof of $\mathtt{NIKE}_{\mathsf{sa}}$ built from a NITE and a TNIKE scheme. The partial maps pks and sks are initially totally undefined.

Proof. In game G_1, the $\mathtt{NITE}$ is operated in the programming mode (with fresh uniformly random values for each programming procedure), while in G_0 the $\mathtt{NITE}$ is operated in the normal mode.

The reduction simulates all the TNIKE parts of the NIKE by itself and obtains the NITE parts from the $\mathsf{G}^{\mathsf{real}}_{\mathtt{NITE}}$ or $\mathsf{G}^{\mathsf{programmed}}_{\mathtt{NITE}}$ game. If $\mathsf{G}^{\mathsf{real}}_{\mathtt{NITE}}$ is played, the reduction simulates G_0, and if $\mathsf{G}^{\mathsf{programmed}}_{\mathtt{NITE}}$ is played, the reduction simulates G_1. □

Lemma 5.10 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$).

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq N_{\mathsf{test}}\varepsilon_{\mathsf{TNIKE}}(\lambda)$$

Proof. In the $\mathcal{O}_{\mathsf{test}}$ queries in game G_1, we compute the shared key with the public key of the first identity and the secret key of the second identity. In game G_2, we ignore the order provided by the adversary and use the larger identity for the public key and the smaller identity for the secret key (with respect to the fixed, public order of the identities).

For each test query in these games, $T^\star$ is neither correlated with $\mathsf{pp}_{\mathsf{TNIKE}}$, nor with the public keys $\mathsf{pk}_{\mathsf{TNIKE},1}$, $\mathsf{pk}_{\mathsf{TNIKE},2}$ or secret keys $\mathsf{sk}_{\mathsf{TNIKE},1}$, $\mathsf{sk}_{\mathsf{TNIKE},2}$ of the queried users. Thus we have

$$\begin{aligned}&\mathtt{SharedKey}_{\mathsf{TNIKE}}(\mathsf{pp}_{\mathsf{TNIKE}}, \mathsf{id}_1^\star, \mathsf{pk}_{\mathsf{TNIKE},1}, \mathsf{id}_2^\star, \mathsf{sk}_{\mathsf{TNIKE},2}, T^\star)\\ &\qquad = \mathtt{SharedKey}_{\mathsf{TNIKE}}(\mathsf{pp}_{\mathsf{TNIKE}}, \mathsf{id}_2^\star, \mathsf{pk}_{\mathsf{TNIKE},2}, \mathsf{id}_1^\star, \mathsf{sk}_{\mathsf{TNIKE},1}, T^\star)\end{aligned}$$

except with probability $\varepsilon_{\mathsf{TNIKE}}(\lambda)$. This proofs that if $\mathsf{id}_1^\star < \mathsf{id}_2^\star$, the games have statistical distance at most $\varepsilon_{\mathsf{TNIKE}}(\lambda)$, and if $\mathsf{id}_2^\star > \mathsf{id}_1^\star$, the games are identical anyway. □

Lemma 5.11 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$).

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq \varepsilon_{\mathsf{hidden}}(\lambda)$$

Proof. In G_3 we no longer program the shared tags of the $\mathtt{NITE}$ to fresh uniformly random values. Instead, we program them to one of the secret tags of the lexicographically larger user (i.e. the user whose $\mathtt{TNIKE}$ secret key is not used in the generation of $K_0^\star$).

The security of the TNIKE guarantees that, for every user i, all of the tags described by the list returned by $\mathtt{SecretTags}(\mathsf{sk}_{\mathsf{TNIKE},i})$ are $\varepsilon_{\mathsf{hidden}}(\lambda)$-close to uniformly random values which are not correlated with the public parameters, any public key or secret key except $\mathsf{sk}_{\mathsf{TNIKE},i}$. Since $\mathsf{sk}_{\mathsf{TNIKE},i}$ is used in G_3 only for the generation of challenge tags $T^\star$, these games are distributed identically except with probability $\varepsilon_{\mathsf{hidden}}(\lambda)$. □

Lemma 5.12 (G_3). *For every PPT adversary $\mathcal{A}$ there exists an PPT adversary $\mathcal{B}$ such that*

$$\Pr[\mathsf{G}_3^{\mathcal{A}}(\lambda) \Rightarrow 1] \leq \mathsf{Adv}^{\nu\text{-semi-adaptive}}_{\mathcal{B},\mathtt{TNIKE}}(\lambda)$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$.

Proof. The reduction simulates all the NITE parts of $\mathtt{NIKE}_{\mathsf{sa}}$ herself and obtains the TNIKE parts from the experiment $\mathsf{Exp}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda)$. This results exactly in the game G_3. □

Combining Lemmata 5.9–5.12 proves Theorem 5.8. □

5.2 Our inner-product-based NIKE as NITE

We prove that $\mathtt{NIKE}_{\mathsf{ip}}$ (Figure 6) can be extended to a ν-programmable NITE with tag space $([\cdot]_T, \mathbb{Z}_q, \mathbb{G}_T)$. The additional algorithms are given in Figure 14. The proof works similar to the one for ν-bounded security, except that the matrix $\mathbf{M}$ from the public parameters is generated on-the-fly.

Theorem 5.13 (Security). *For every PPT adversary $\mathcal{A}$ for the NITE from Figures 6 and 14, there exists a PPT adversary $\mathcal{B}$ for the $\mathcal{U}_k$-*MDDH *with*

$$\mathsf{Adv}^{\nu\text{-programmable}}_{\mathcal{A},\mathtt{NITE}}(\lambda) \leq \mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B},\mathcal{D}_{\ell,k},\mathtt{SymGGen},s}(\lambda) + \frac{1}{q-1}$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ for a polynomial poly *independent of $\mathcal{A}$.*

```
$\texttt{Setup}'(1^\lambda)$:
    $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \texttt{SymGGen}(1^\lambda)$
    $\mathbf{D} \leftarrow \mathcal{U}_{k+\nu,k}$
    $\mathbf{M} \xleftarrow{\$} \mathsf{Sym}_{k+\nu}(\mathbb{Z}_q)$
    $\mathsf{pp} := (\mathcal{G}, [\mathbf{D}], [\mathbf{MD}])$
    $\mathsf{us} : \mathcal{IDS} \dashrightarrow \mathbb{Z}_q^{k+\nu}$
    $st := (\mathcal{G}, \mathbf{D}, \mathbf{M}, \mathsf{us})$
    return $(\mathsf{pp}, st)$

$\texttt{RegH}(st, \mathsf{id})$:
    parse $st =: (\mathcal{G}, \mathbf{D}, \mathbf{M}, \mathsf{us})$
    $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{k+\nu}$
    $\mathsf{pk} := [\mathbf{u}]$
    $\mathsf{us}(\mathsf{id}) := \mathbf{u}$
    $st := (\mathcal{G}, \mathbf{D}, \mathbf{M}, \mathsf{us})$
    return $(\mathsf{pk}, st)$
```

```
$\texttt{RevH}(st, \mathsf{id}_1\mathsf{id}_2)$:
    parse $st =: (\mathcal{G}, \mathbf{D}, \mathbf{M}, \mathsf{us})$
    if $\mathsf{us}(\mathsf{id}_1) = \bot \vee \mathsf{us}(\mathsf{id}_2) = \bot$ then
        $K := \bot$
    else
        $\mathbf{u}_1 := \mathsf{us}(\mathsf{id}_1)$;  $\mathbf{u}_2 := \mathsf{us}(\mathsf{id}_2)$
        $K := \mathbf{u}_1^\top \mathbf{M} \mathbf{u}_2$
    return $st, K$

$\texttt{Extr}(st, \mathsf{id})$:
    parse $st =: (\mathcal{G}, \mathbf{D}, \mathbf{M}, \mathsf{us})$
    if $\mathsf{us}(\mathsf{id}) = \bot$ then
        $\mathsf{sk} := \bot$
    else
        $\mathsf{sk} := \mathbf{M} \cdot \mathsf{us}(\mathsf{id})$
    return $st, \mathsf{sk}$
```

```
$\texttt{Test}(st, \mathsf{id}_1^\star, \mathsf{id}_2^\star, h_\ell^\star \in \mathcal{H})$:
    parse $st =: (\mathcal{G}, \mathbf{D}, \mathbf{M}, \mathsf{us})$
    if $\mathsf{us}(\mathsf{id}_1^\star) = \bot \vee \mathsf{us}(\mathsf{id}_2^\star) = \bot$ then
        compute $\mathbf{x} \in \mathbb{Z}_q^{k+\nu} \setminus \{\mathbf{0}\}$ such that $\mathbf{D}^\top \mathbf{x} = \mathbf{0} \wedge \forall \mathsf{id} \in Q_{\mathsf{inv}} \setminus \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\} : \mathsf{us}(\mathsf{id})^\top \mathbf{x} = 0$
        $\mathbf{u}_1 := \mathsf{us}(\mathsf{id}_1^\star)$;  $\mathbf{u}_2 := \mathsf{us}(\mathsf{id}_2^\star)$
        if $\mathsf{us}(\mathsf{id}_1^\star)^\top \mathbf{x} = 0 \vee \mathsf{us}(\mathsf{id}_2^\star)^\top \mathbf{x} = 0$ then abort
        $\mu := \frac{h_\ell^\star - \mathbf{u}_1^\top \mathbf{M} \mathbf{u}_2}{\mathbf{u}_1^\top \mathbf{x} \cdot \mathbf{u}_2^\top \mathbf{x}}$
        $\mathbf{M} := \mathbf{M} + \mu \mathbf{x} \mathbf{x}^\top$
    $st := (\mathcal{G}, \mathbf{D}, \mathbf{M}, \mathsf{us})$
    return $st$
```

Figure 14: The additional algorithms that turn our core NIKE from Figure 6 into a NITE.

```
$\mathsf{G}_0$ $\boxed{\mathsf{G}_1 \boxed{\mathsf{G}_2}}$

$\mathsf{G}^{\mathsf{real}}_{\mathcal{A},\mathtt{NITE}} / \mathsf{G}^{\mathsf{programmed}}_{\mathcal{A},\mathtt{NITE}}$:
    $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathtt{SymGGen}(1^\lambda)$
    $\mathbf{D} \leftarrow \mathcal{U}_{k+\nu,k}$
    $\mathbf{M} \xleftarrow{\$} \mathsf{Sym}_{k+\nu}(\mathbb{Z}_q)$
    $\mathsf{pp} := (\mathcal{G}, [\mathbf{D}], [\mathbf{MD}])$
    $Q_{\mathsf{extr}} := \emptyset;\ Q_{\mathsf{test}} := \emptyset;\ Q_{\mathsf{inv}} := \emptyset;\ Q_{\mathsf{reg}} := \emptyset$
    $\mathsf{us} : \mathcal{IDS} \dashrightarrow \mathbb{Z}_q^{k+\nu}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot), \mathcal{O}_{\mathsf{extr}}(\cdot), \mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{id} \in Q_{\mathsf{reg}}$ then return $\perp$
    $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^k;\ \mathbf{u} := \mathbf{Dw}$
    $\boxed{\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{k+\nu}}$
    $\mathsf{us}(\mathsf{id}) := \mathbf{u}$
    return $\mathsf{pk} := [\mathbf{u}]$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{id} \notin Q_{\mathsf{reg}}$ then return $\perp$
    $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\};\ Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$
    return $[\mathbf{M} \cdot \mathsf{us}(\mathsf{id})]$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{id}_1^\star, \mathsf{id}_2^\star \in Q_{\mathsf{reg}}$ then
        $Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$
        $T^\star := \mathsf{us}(\mathsf{id}_1^\star)^\top \mathbf{M} \mathsf{us}(\mathsf{id}_2^\star)$
            if $|Q_{\mathsf{inv}}| \leq \nu \wedge \mathsf{id}_1^\star, \mathsf{id}_2^\star \notin Q_{\mathsf{extr}} \wedge \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\} \notin Q_{\mathsf{test}}$ then
                $h_\ell^\star \xleftarrow{\$} \mathbb{Z}_q;\ T^\star := [h_\ell^\star]_T$
                compute $\mathbf{x} \in \mathbb{Z}_q^{k+\nu} \setminus \{\mathbf{0}\}$ such that $\mathbf{D}^\top \mathbf{x} = \mathbf{0} \wedge \forall \mathsf{id} \in Q_{\mathsf{inv}} \setminus \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\} : \mathsf{us}(\mathsf{id})^\top \mathbf{x} = 0$
                $\mathbf{u}_1 := \mathsf{us}(\mathsf{id}_1^\star);\ \mathbf{u}_2 := \mathsf{us}(\mathsf{id}_2^\star)$
                if $\mathsf{us}(\mathsf{id}_1^\star)^\top \mathbf{x} = 0 \vee \mathsf{us}(\mathsf{id}_2^\star)^\top \mathbf{x} = 0$ then return $\perp$
                $\mu := \frac{h_\ell^\star - \mathbf{u}_1^\top \mathbf{M} \mathbf{u}_2}{\mathbf{u}_1^\top \mathbf{x} \cdot \mathbf{u}_2^\top \mathbf{x}}$
                $\mathbf{M} := \mathbf{M} + \mu \mathbf{x}\mathbf{x}^\top$
            $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\}$
        return $T^\star$
    return $\perp$
```

Figure 15: Hybrids for the ν-programmability of the NITE from Figures 6 and 14. The partial map us is initially totally undefined and stores the users public keys in the exponent.

Proof. The proof uses a hybrid argument with hybrids G_0–G_2 given in Figure 15. The game G_2 implicitly defines the algorithms $\mathtt{Setup}'$, $\mathtt{RegH}$, $\mathtt{Test}$, $\mathtt{RevH}$, and $\mathtt{Extr}$ for our NITE.

Lemma 5.14 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). *For every PPT adversary $\mathcal{A}$ there exists an PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{mddh}}_{\mathcal{B},\mathcal{U}_{k+\nu,k},\mathtt{SymGGen}}(\lambda) + \frac{1}{q-1}$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ for a polynomial poly *independent of $\mathcal{A}$.*

Proof. In the real game G_0, the users public keys are chosen from the linear span of $\mathbf{B}$, while in G_1, they are chosen from the full space $\mathbb{Z}_q^{k+\nu}$.

Given an N-fold $\mathcal{U}_k$-MDDH challenge $[\mathbf{D}], ([\mathbf{u}_i])_{1\leq i\leq N}$ one can simulate the game G_0 or G_1 efficiently when $\mathbf{A}$ is known over $\mathbb{Z}_q$. If the vectors $\mathbf{u}_i$ are sampled from the linear span of $\mathbf{D}$ this yields the game G_0 and if the vectors $\mathbf{u}_i$ are sampled uniformly random, this yields the game G_1. □

Lemma 5.15 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$). *For every PPT adversary $\mathcal{A}$ there exists an PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq \frac{2}{q}$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ for a polynomial poly *independent of $\mathcal{A}$.*

Proof. In G_2 the $\mathcal{O}_{\mathsf{test}}$ queries are answered with tags programmed by the $\mathtt{Test}$ algorithm as longs as this is possible.

When the adversary issues a test query on $\mathsf{id}_1^\star$ and $\mathsf{id}_2^\star$ with $|Q_{\mathsf{inv}}| \leq \nu$, $\mathsf{id}_1^\star, \mathsf{id}_2^\star \notin Q_{\mathsf{extr}}$, and $\{\{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}\} \notin Q_{\mathsf{test}}$, the query will be answered with a programmed tag in G_2. Let us assume that $\mathsf{us}(\mathsf{id}_1^\star)^\top \mathbf{x} \neq 0$ and $\mathsf{us}(\mathsf{id}_2^\star)^\top \mathbf{x} \neq 0$. This happens when $\mathsf{us}(\mathsf{id}_1^\star)$ and $\mathsf{us}(\mathsf{id}_2^\star)$ are both not in the linear span of $\mathbf{D}$ and the public keys of all other users that where involved in an extract or test query before. Since $|Q_{\mathsf{inv}}| \leq \nu$, this "forbidden subspace" has dimension at most $k+\nu-2$ and thus the event described above occurs with probability at least $1-2/q$. Then, in G_2, we replace on-the-fly the matrix $\mathbf{M}$ by $\mathbf{M} + \mu\mathbf{x}\mathbf{x}^\top$. However, when $\mathbf{M}$ is a uniformly random symmetric matrix, then so is $\mathbf{M} + \mu\mathbf{x}\mathbf{x}^\top$. This change is consistent with the public parameters, because

$$(\mathbf{M} + \mu\mathbf{x}\mathbf{x}^\top)\mathbf{D} = \mathbf{M}\mathbf{D} + \mu\mathbf{x}\underbrace{\mathbf{x}^\top\mathbf{D}}_{=\mathbf{0}} = \mathbf{M}\mathbf{D}.$$

The change of $\mathbf{M}$ is also consistent with the secret keys revealed so far, because when the secret key of user id is revealed, $\mathsf{id} \notin \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$ and thus $\mathbf{x}^\top \mathsf{us}(\mathsf{id}^\star) = \mathbf{0}$ and the revealed secret key for this user is

$$\mathbf{M}\mathsf{us}(\mathsf{id}) = (\mathbf{M} + \mu\mathbf{x}\mathbf{x}^\top)\mathsf{us}(\mathsf{id}).$$

And when the reduction revealed the shared key $\mathsf{us}(\mathsf{id}_1)\mathbf{M}\mathsf{us}(\mathsf{id}_2)$ of the users id_1 and id_2, then for at least one $i \in \{1,2\}$ $\mathsf{id}_i \notin \{\mathsf{id}_1^\star, \mathsf{id}_2^\star\}$ and thus $\mathbf{x}^\top \mathsf{us}(\mathsf{id}_i)$ and

$$\mathsf{us}(\mathsf{id}_1)^\top \mathbf{M}\mathsf{us}(\mathsf{id}_2) = \mathsf{us}(\mathsf{id}_1)^\top(\mathbf{M} + \mu\mathbf{x}\mathbf{x}^\top)\mathsf{us}(\mathsf{id}_2).$$

Finally, by the definition of μ, we have

$$h_\ell^\star = \mathsf{us}(\mathsf{id}_1^\star)^\top \mathbf{M}\mathsf{us}(\mathsf{id}_2^\star) + \mu\mathsf{us}(\mathsf{id}_1^\star)^\top \mathbf{x}\mathsf{us}(\mathsf{id}_2^\star)^\top \mathbf{x} = \mathsf{us}(\mathsf{id}_1^\star)^\top(\mathbf{M} + \mu\mathbf{x}\mathbf{x}^\top)\mathsf{us}(\mathsf{id}_2^\star) = \mathsf{us}(\mathsf{id}_1^\star)^\top \mathbf{M}\mathsf{us}(\mathsf{id}_2^\star)$$

and thus $T^\star$ is a valid answer to the test query in G_2. □

Combining Lemmata 5.14 and 5.15 shows Theorem 5.13. □

5.3 A TNIKE from LWE

We give a construction of a ν-semi-adaptively secure TNIKE in Figure 16. It has key space $\mathcal{K} = \{0,1\}^\mu$ for an arbitrary, polynomial $\mu \in \mathbb{N}_+$ and needs any tag space $(f, \mathcal{H}, \mathcal{T})$ with $\mathcal{T} \subseteq \{0,1\}^r$ with superpolynomial cardinality and polynomial $r \in \mathbb{N}_+$. In particular, $\mathcal{T} = \mathbb{G}_T$ works with an appropriate binary

```
$\mathsf{Setup}(1^\lambda)$:
    $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{n \times m}$
    return $\mathsf{pp} := \mathbf{A}$

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{id})$:
    parse $\mathsf{pp} =: \mathbf{A}$
    $\mathbf{t} \xleftarrow{\$} \{0,1\}^m$
    $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{\mu \times n}$
    $\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z}_p^{\mu \times m},\sigma}$
    $\mathsf{pk}_{\mathrm{I}} := \mathbf{SA} + \mathbf{E}$
    $\mathsf{sk}_{\mathrm{I}} := \mathbf{S}$
    for $\ell \in \{1, \ldots, \nu\}$ do
        $h_\ell \xleftarrow{\$} \mathcal{H}$; $\tau_\ell := f(h_\ell)$
        for $k \in \{1, \ldots, r\}$ do
            $\mathbf{U}_{\ell,k} \xleftarrow{\$} [-1,1]^{m \times m}$
            $\mathbf{V}_{\ell,k} := \mathbf{AU}_{\ell,k} + \tau_\ell[k]\mathbf{G}$
    $\mathsf{pk}_{\mathrm{II}} := ((\mathbf{V}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r}, \mathbf{t})$
    $\mathsf{sk}_{\mathrm{II}} := ((h_\ell)_{1 \le \ell \le \nu}, (\mathbf{U}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r}, \mathbf{t})$
    $\mathsf{pk} := (\mathsf{pk}_{\mathrm{I}}, \mathsf{pk}_{\mathrm{II}})$
    $\mathsf{sk} := (\mathsf{sk}_{\mathrm{I}}, \mathsf{sk}_{\mathrm{II}})$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{SecretTags}(\mathsf{sk})$:
    parse $\mathsf{sk} =: (\_, ((h_\ell)_{1 \le \ell \le \nu}, \_, \_))$
    return $(h_\ell)_{1 \le \ell \le \nu}$
```

```
$\mathsf{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2, T)$:
    define $g_T : (\{0,1\}^r)^\nu \to \{0,1\}$
                $(\tau_1, \ldots, \tau_\nu) \mapsto (\exists \ell \in \{1, \ldots, \nu\} : \tau_\ell \overset{?}{=} T)$
    if $\mathsf{id}_1 < \mathsf{id}_2$ then
        parse $\mathsf{sk}_2 =: (\_, ((h_\ell)_{1 \le \ell \le \nu}, (\mathbf{U}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r}, \mathbf{t}))$
        if $\ell \in \{1, \ldots, \nu\}$ then $\tau_\ell := f(h_\ell)$
        if $\exists \ell \in \{1, \ldots, \nu\} : \tau_\ell = T$ then
            return $\perp$
        else
            $\mathbf{U}^\star := \mathsf{Eval}^{\mathsf{in}}(g_T, (\tau_\ell[k], \mathbf{U}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r})$
            parse $\mathsf{pk}_1 =: (\mathbf{R}, \_)$
            return $\mathsf{Round}(\mathbf{RU}^\star\mathbf{t})$
    else
        parse $\mathsf{pk}_1 =: (\_, ((\mathbf{V}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r}, \mathbf{t}))$
        parse $\mathsf{sk}_2 =: (\mathbf{S}, \_)$
        $\mathbf{V}^\star := \mathsf{Eval}^{\mathsf{out}}(g_T, (\mathbf{V}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r})$
        return $\mathsf{Round}(\mathbf{SV}^\star\mathbf{t})$
```

where

$$\begin{aligned} \mathsf{Round} : \mathbb{Z}_p^\mu &\to \{0,1\}^\mu \\ (x_1, \ldots, x_\mu)^\top &\mapsto (\mathsf{round}(x_1), \ldots, \mathsf{round}(x_\mu))^\top \\ \mathsf{round} : \mathbb{Z}_p &\to \{0,1\} \\ x &\mapsto \begin{cases} 0 & \text{if } x \in [0, (p-1)/2] \\ 1 & \text{if } x \in [(p-1)/2 + 1, p-1] \end{cases} \end{aligned}$$

Figure 16: Our construction of a ν-semi-adaptively secure TNIKE. Recall that $\tau_\ell[k]$ stands for the k-th bit of the binary representation of τ_ℓ

representation of group elements. We assume that there is a total order on the identity space known to all users.

This construction is based on the leveled fully homomorphic trapdoor functions by Gorbunov et al. [25] in a non-blackbox way. This function is given by $f(\tau_\ell[k] \in \{0,1\}) = \mathbf{AU}_{\ell,k} + \tau_\ell[k]\mathbf{G}$, where $\mathbf{A}$ is a fixed uniformly random matrix, $\mathbf{U}_{\ell,k}$ is a short random matrix and $\mathbf{G} := \mathbf{I}_n \otimes (2^0, \ldots, 2^{\lceil \log p + 1 \rceil - 1})$. Now given for $k \in \{1, \ldots, r\}$ the values $\tau_\ell[k] \in \{0,1\}$, $\mathbf{U}_{\ell,k} \in [-1,1]^{m \times m}$, $\mathbf{V}_{\ell,k} := \mathbf{AU}_{\ell,k} + \tau_\ell[k]\mathbf{G}$ and a function $g : \{0,1\}^r \to \{0,1\}$ we can use the algorithms $\mathsf{Eval}^{\mathsf{in}}$ and $\mathsf{Eval}^{\mathsf{out}}$ to compute a short matrix $\mathbf{U}^\star := \mathsf{Eval}^{\mathsf{in}}(g, (\tau_\ell[k], \mathbf{U}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r})$ and $\mathbf{V}^\star := \mathsf{Eval}^{\mathsf{out}}(g, (\mathbf{V}_{\ell,k})_{1 \le \ell \le \nu, 1 \le k \le r})$ with $\mathbf{V}^\star = \mathbf{AU}^\star + g((\tau_\ell[k])_{1 \le k \le r})\mathbf{G}$. Details on these functions and the noise growth in $\mathsf{Eval}^{\mathsf{in}}$ can be found in [25].

Our TNIKE construction takes as parameters $\ell, n, m \in \mathbb{N}_+$ that grow polynomially in λ, a prime p with polynomial bit length, $\gamma, \sigma_1, \sigma, \tilde{\sigma}, \sigma', \varepsilon \in \mathbb{R}_+$ with the following constraints

1. The $(\ell, m, p, \mathcal{D}_{\mathbb{Z}_p^n,\gamma})$-LWE assumption is hard
2. $\frac{p^{\ell+1}}{\sigma_1^n}$ is negligible
3. $m = n\lceil \log p + 1 \rceil$
4. $\sigma_1 \ge \sqrt{\ln(2\mu m(1 + 1/\varepsilon)/\pi}$ for a negligible ε
5. $\tilde{\sigma} > C\sqrt{m}\gamma\sigma_1$ for the constant C form Lemma 2.12
6. $\sigma^2 = \sigma_1^2 + \tilde{\sigma}^2$
7. $p/\sigma_1 \ge \sqrt{\ln(4n)/\pi}$

To see that that all these constraints can be satisfied, observe the following. Constraint 1 is conjectured to be satisfiable for appropriate choices of ℓ, p, γ, independent of m. Constraint 2 can be satisfied by picking n large enough (and some very mild constraint on σ_1, e.g., $\sigma_1 \geq 2$). The constraints 3–6 each contain one variable that has not been determined by the constraints before and thus are easy to satisfy. The last constraint is a technical requirement that already appears in [8]. It is satisfied automatically by all realistic parameter choices.

Theorem 5.16 (Correctness). *The TNIKE in Figure 16 is statistically correct with correctness error*

$$\mathcal{O}\left(\frac{\nu m^2 r \mu \sigma \sqrt{\ln(p/\sigma)}}{p} + \frac{\nu}{|\mathcal{T}|}\right).$$

Proof. Let $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$, $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id}_1)$, and $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathtt{KeyGen}(\mathsf{pp}, \mathsf{id}_2)$. Without loss of generality, assume $\mathsf{id}_1 < \mathsf{id}_2$. Also assume that $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2, T) \neq \perp$. This happens with probability $1 - \frac{\nu}{|\mathcal{T}|}$, because each of the value h_ℓ ($\ell \in \{1, \dots, \nu\}$) of user id_2's secret key and thus also $\tau_\ell := f(h_\ell)$ are uniformly random and independent of T.

Under these assumptions,

$$\begin{aligned}&\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2, T)\\ &\qquad = \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, (\mathbf{SA} + \mathbf{E}, \_), \mathsf{id}_2, (\_, ((h_\ell)_{1 \leq \ell \leq \nu}, (\mathbf{U}_{\ell,k})_{1 \leq \ell \leq \nu, 1 \leq k \leq r}, \mathbf{t})), T)\\ &\qquad\qquad = \mathsf{Round}(\mathbf{SAU}^\star \mathbf{t} + \mathbf{EU}^\star \mathbf{t})\end{aligned}$$

for $\mathbf{U}^\star := \mathsf{Eval}^{\mathsf{in}}(g_T, (\tau_\ell[k], \mathbf{U}_{\ell,k})_{1 \leq \ell \leq \nu, 1 \leq k \leq r})$ and

$$\begin{aligned}\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_2, \mathsf{pk}_2, \mathsf{id}_1, \mathsf{sk}_1, T) &= \mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_2, (\_, ((\mathbf{V}_{\ell,k})_{1 \leq \ell \leq \nu, 1 \leq k \leq r}, \mathbf{t})), \mathsf{id}_2, (\mathbf{S}, \_), T)\\ &= \mathsf{Round}(\mathbf{SV}^\star \mathbf{t}) = \mathsf{Round}(\mathbf{SAU}^\star \mathbf{t})\end{aligned}$$

for $\mathbf{V}^\star := \mathsf{Eval}^{\mathsf{out}}(g_T, (\mathbf{V}_{\ell,k})_{1 \leq \ell \leq \nu, 1 \leq k \leq r})$. Thus the keys are identical if the error term $\mathbf{EU}^\star \mathbf{t}$ is removed by the rounding operation.

The circuit for the function g_T first does for each $\ell \in \{1, \dots, \nu\}$ a bit-wise equality test with the constant T and τ_ℓ. This does not increase the noise. Then the bit-wise results are multiplied with a chain of multiplication gates. This amplifies the noise by a factor of mr. Then these results for each $\ell \in \{1, \dots, \nu\}$ are added together, which also means the noise terms are added together. Since for all $k \in \{1, \dots, r\}$, we have $\|\mathbf{U}_{\ell,k}\|_\infty \leq 1$, we have $\|\mathbf{U}^\star\|_\infty \leq mr$ and thus $\|\mathbf{U}^\star \mathbf{t}\|_\infty \leq \nu m^2 r$. Now assume that $\|\mathbf{E}\|_\infty \leq \sigma\sqrt{2\ln(p/\sigma)}$. By Lemma 2.8, each entry $\mathbf{E}_{\tilde{\imath},\tilde{\jmath}}$ of $\mathbf{E}$ satisfies

$$\Pr[|\mathbf{E}_{\tilde{\imath},\tilde{\jmath}}| \geq \sigma\sqrt{2\ln(p/\sigma)}] \leq 2e^{-\frac{\left(\sigma\sqrt{2\ln(p/\sigma)}\right)^2}{2\sigma}} = 2e^{-\ln(p/\sigma)} = \frac{2\sigma}{p}.$$

Thus $\|\mathbf{E}\|_\infty \leq \sigma\sqrt{2\ln(p/\sigma)}$ occurs with probability at least $1 - 2\sigma\mu m/p$ and in this case $\|\mathbf{EU}^\star \mathbf{t}\|_\infty \leq \nu m^2 r \sigma\sqrt{2\ln(p/\sigma)}$.

Now, if all entries of $\mathbf{SAU}^\star \mathbf{t}$, which each have statistical distance at most $2^{-m} + p^{-m} + p^{-1}$ from a uniformly random variable, are not within distance $m^2 r \sigma\sqrt{2\ln(p/\sigma)}$ of one of the two "rounding points" 0 and $(p-1)/2$, the error term is removed by the rounding operation. The later occurs with probability

$$\left(\frac{4\nu m^2 r \sigma\sqrt{2\ln(p/\sigma)} - 3}{p} + 2^{-m} + p^{-m} + p^{-1}\right)\mu,$$

which leads to a total correctness error of

$$\left(\frac{4\nu m^2 r \sigma\sqrt{2\ln(p/\sigma)} + 2\sigma m - 2}{p} + 2^{-m} + p^{-m}\right)\mu + \frac{\nu}{|\mathcal{T}|}. \qquad \square$$

Theorem 5.17 (Security). *The TNIKE from Figure 16 enjoys ν-semi-adaptive security. In particular, for every PPT adversary $\mathcal{A}$ for this TNIKE, there exists a PPT adversary $\mathcal{B}$ for the $(\ell, m, p, \mathcal{D}_{\mathbb{Z}_p^n,\gamma})$-*LWE *problem with*

$$\mathsf{Adv}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda) \leq 2m\mathsf{Adv}^{\mathsf{lwe}}_{n,m,p,\mathcal{D}_{\mathbb{Z}_p^n},\gamma}(\mathcal{B}) + 16\varepsilon N + 2^{1-m} + \sqrt{2}\mu\nu\sqrt{\frac{p^{\ell+1}\nu}{\sigma_1^n}}$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$ *for a polynomial* poly *independent of* $\mathcal{A}$.

Proof. First of all, we show that all the tag descriptions $(h_\ell)_{1\leq\ell\leq\nu}$ returned by $\mathtt{SecretTags}(\mathsf{sk})$ are independent of the corresponding public parameters and public key with overwhelming probability. Only the values $\mathbf{V}_{\ell,k} = \mathbf{A}\mathbf{U}_{\ell,k} + \tau_\ell[k]\mathbf{G}$ in the public key depend on h_ℓ. However, since multiplication with the uniformly random matrix $\mathbf{A}$ is a universal hash function and $\mathbf{U}_{\ell,k}$ contains $m^2\log 3$ bits of min entropy, the LoHL (Lemma 2.18) guarantees that each $\mathbf{V}_{\ell,k}$ has statistical distance at most

$$\frac{1}{2}\sqrt{2^{-m^2\log 3}nm\log p} \leq \frac{1}{2}\sqrt{(2/3)^{m^2}}$$

to a uniformly random value and when all $\mathbf{V}_{\ell,k}$ would be uniformly random, h_ℓ would be perfectly hidden. This leaves us with a tag hiding error of

$$\varepsilon_{\mathsf{hidden}}(\lambda) = \frac{\nu r}{2}\sqrt{(2/3)^{m^2}}.$$

Next, we prove that $\mathsf{Adv}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda)$ is negligible. The proof uses a hybrid argument with hybrids G_0–G_4 given in Figure 17. Some hybrids perform computations with real numbers, but these hybrids are only involved in information-theoretic transitions. Thus we do not need to worry about precision or computability of these operations.

Lemma 5.18 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). *For every PPT adversary $\mathcal{A}$ there exists an PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq m\mathsf{Adv}^{\mathsf{lwe}}_{n,m,p,\mathcal{D}_{\mathbb{Z}_p^n},\gamma}(\mathcal{B})$$

and $T(\mathcal{B}) \approx T(\mathcal{A}) + N\,\mathrm{poly}(\lambda)$.

Proof. In G_1 the matrix $\mathbf{A}$ is in "lossy mode", that is, $\mathbf{A}$ is a low-rank matrix plus small noise. Under the LWE-assumption, this is indistinguishable form a uniformly random matrix. For this transition we need m LWE instances on the matrix $\mathbf{B}^\top$, that is, a tuple $(\mathbf{B}^\top, (\mathbf{a}_i)_{1\leq i\leq m})$ where either $\mathbf{a}_i = \mathbf{B}\mathbf{c}_i + \mathbf{z}_i$ for $\mathbf{c}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $\mathbf{z}_i \leftarrow \mathcal{D}_{\mathbb{Z}_p^n,\gamma}$ or $\mathbf{a}_i \xleftarrow{\$} \mathbb{Z}_p^n$. With a hybrid argument, one can show that these two distributions are indistinguishable under the $(\ell, m, p, \mathcal{D}_{\mathbb{Z}_p^n,\gamma})$-LWE assumption with security loss m. The reduction between G_0 and G_1 sets $\mathbf{A} = (\mathbf{a}_1|\cdots|\mathbf{a}_m)$ and simulates the remaining part honestly. If the vectors $\mathbf{a}_i$ are uniformly random, the reduction simulates $\mathbf{a}_i$ and if they are of the form $\mathbf{a}_i = \mathbf{B}\mathbf{c}_i + \mathbf{z}_i$, the reduction simulates G_1. □

Lemma 5.19 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$).

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq 8\varepsilon N$$

Proof. In G_1, the noise matrix $\mathcal{D}_{\mathbb{Z}_p^{\mu\times m},\sigma}$ for the public keys is sampled from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_p^{\mu\times m},\sigma}$. In G_2 however it is sampled with the randomized rounding procedure of [40]. That is, we sample first $\widetilde{\mathbf{E}}$ from the continuous Gaussian distribution $\mathcal{C}^{\mu\times m}_{\tilde{\sigma},p}$, then we sample $\mathbf{E}'$ from the discrete Gaussian $\mathcal{D}_{\mathbb{Z}_p^{\mu\times m}-\widetilde{\mathbf{E}},\sigma'}$ and set $\mathbf{E} := \widetilde{\mathbf{E}} + \mathbf{E}'$.

The parameter constraints 4 and 6 allow us to apply Lemma 2.9. This shows that each public key in G_2 is within statistical distance of 8ε to the public key in G_1. □

Lemma 5.20 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$).

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}}(\lambda) \Rightarrow 1]\right| \leq 2^{-m}$$

G_0 G_1 G_2 G_3 G_4

```
$\mathsf{Exp}^{\nu\text{-semi-adaptive}}_{\mathcal{A},\mathtt{TNIKE}}(\lambda)$:
    $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{n\times m}$
    $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{n\times \ell}$; $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times m}$; $\mathbf{Z} \leftarrow \mathcal{D}_{\mathbb{Z}_p^{n\times m},\gamma}$
    $\mathbf{A} := \mathbf{BC} + \mathbf{Z}$
    $\mathsf{pp} := \mathbf{A}$
    $Q_{\mathsf{extr}} := \emptyset$; $Q_{\mathsf{rev}} := \emptyset$; $Q_{\mathsf{test}} := \emptyset$; $Q_{\mathsf{inv}} := \emptyset$
    $\mathsf{pks} : \mathcal{IDS} \dashrightarrow \mathcal{PK}$
    $\mathsf{sks} : \mathcal{IDS} \dashrightarrow \mathcal{SK}$
    $b \xleftarrow{\$} \{0,1\}$
    $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{regH}}(\cdot),\mathcal{O}_{\mathsf{extr}}(\cdot),\mathcal{O}_{\mathsf{revH}}(\cdot,\cdot),\mathcal{O}_{\mathsf{test}}(\cdot,\cdot)}(\mathsf{pp})$
    if $Q_{\mathsf{rev}} \cap Q_{\mathsf{test}} = \emptyset \land \nexists A \in Q_{\mathsf{test}} : A \cap Q_{\mathsf{extr}} = \emptyset$
    then
        return $b \overset{?}{=} b^\star$
    else
        return $0$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{id} \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}) \neq \bot$ then return $\bot$
    $\mathbf{t} \xleftarrow{\$} \{0,1\}^m$
    $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{\mu\times n}$
    $\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z}_p^{\mu\times m},\sigma}$
    $\widetilde{\mathbf{E}} \xleftarrow{\$} \mathcal{C}^{\mu\times m}_{\tilde{\sigma},p}$
    $\Sigma := \tilde{\sigma}^2\mathbf{I}_m - \sigma_1^2\mathbf{Z}^\top\mathbf{Z}$
    $\mathbf{E}_1 \leftarrow \mathcal{C}^{\mu\times n}_{\sigma_1,p}$; $\mathbf{E}_2 \leftarrow \mathcal{C}^{\mu\times m}_{\Sigma,p}$
    $\widetilde{\mathbf{E}} := \mathbf{E}_1\mathbf{Z} + \mathbf{E}_2$
    $\mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z}_p^{\mu\times m}-\widetilde{\mathbf{E}},\sigma'}$; $\mathbf{E} := \widetilde{\mathbf{E}} + \mathbf{E}'$
    $\mathsf{pk}_{\mathrm{I}} := \mathbf{SA} + \mathbf{E}$
    $\mathsf{sk}_{\mathrm{I}} := \mathbf{S}$
    for $\ell \in \{1,\dots,\nu\}$ do
        $h_\ell \xleftarrow{\$} \mathcal{H}$; $\tau_\ell := f(h_\ell)$
        for $k \in \{1,\dots,r\}$ do
            $\mathbf{U}_{\ell,k} \xleftarrow{\$} [-1,1]^{m\times m}$
            $\mathbf{V}_{\ell,k} := \mathbf{AU}_{\ell,k} + \tau_\ell[k]\mathbf{G}$
    $\mathsf{pk}_{\mathrm{II}} := ((\mathbf{V}_{\ell,k})_{1\le\ell\le\nu,1\le k\le r}, \mathbf{t})$
    $\mathsf{sk}_{\mathrm{II}} := ((h_\ell)_{1\le\ell\le\nu}, (\mathbf{U}_{\ell,k})_{1\le\ell\le\nu,1\le k\le r}, \mathbf{t})$
    $\mathsf{pk} := (\mathsf{pk}_{\mathrm{I}}, \mathsf{pk}_{\mathrm{II}})$
    $\mathsf{sk} := (\mathsf{sk}_{\mathrm{I}}, \mathsf{sk}_{\mathrm{II}})$
    $\mathsf{pks}(\mathsf{id}) := \mathsf{pk}$; $\mathsf{sks}(\mathsf{id}) := \mathsf{sk}$
    return $\mathsf{pk}$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{id} \in \mathcal{IDS})$:
    $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{id}\}$; $Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}\}$
    return $\mathsf{sks}(\mathsf{id})$

$\mathcal{O}_{\mathsf{revH}}(\mathsf{id}_1 \in \mathcal{IDS}, \mathsf{id}_2 \in \mathcal{IDS}, T \in \mathcal{T})$:
    if $\mathsf{pks}(\mathsf{id}_1) \neq \bot \land \mathsf{sks}(\mathsf{id}_2) \neq \bot$ then
        $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{id}_1,\mathsf{id}_2\}\}$; $Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1,\mathsf{id}_2\}$
        $\mathsf{pk}_1 := \mathsf{pks}(\mathsf{id}_1)$; $\mathsf{sk}_2 := \mathsf{sks}(\mathsf{id}_2)$
        return $\mathtt{SharedKey}(\mathsf{pp}, \mathsf{id}_1, \mathsf{pk}_1, \mathsf{id}_2, \mathsf{sk}_2, T)$
    return $\bot$

$\mathcal{O}_{\mathsf{test}}(\mathsf{id}_1^\star \in \mathcal{IDS}, \mathsf{id}_2^\star \in \mathcal{IDS})$:
    if $\mathsf{pks}(\mathsf{id}_1^\star) \neq \bot \land \mathsf{sks}(\mathsf{id}_2^\star) \neq \bot \land \{\mathsf{id}_1^\star,\mathsf{id}_2^\star\} \notin Q_{\mathsf{test}}$
    then
        $Q_{\mathsf{test}} := Q_{\mathsf{test}} \cup \{\{\mathsf{id}_1^\star,\mathsf{id}_2^\star\}\}$; $Q_{\mathsf{inv}} := Q_{\mathsf{inv}} \cup \{\mathsf{id}_1^\star,\mathsf{id}_2^\star\}$
        if $|Q_{\mathsf{inv}}| < \nu$ then
            $(i,j) := (\arg\max_{i\in\{1,2\}} \mathsf{id}_i^\star, \arg\min_{i\in\{1,2\}} \mathsf{id}_i^\star)$
            $\mathsf{pk} := \mathsf{pks}(\mathsf{id}_i^\star)$; $\mathsf{sk} := \mathsf{sks}(\mathsf{id}_j^\star)$
            $h_\ell^\star \leftarrow \mathtt{SecretTags}(\mathsf{sk})$; $T^\star := f(h_\ell^\star)$
            define $g_{T^\star} : (\{0,1\}^r)^\nu \to \{0,1\}$
                $(\tau_1,\dots,\tau_\nu) \mapsto (\exists \ell \in \{1,\dots,\nu\} : \tau_\ell \overset{?}{=} T^\star)$
            parse $\mathsf{pk}_i =: (\_, ((\mathbf{V}_{\ell,k})_{1\le\ell\le\nu,1\le k\le r}, \mathbf{t}))$
            parse $\mathsf{sk}_i =: (\mathbf{S}, \_)$
            $\mathbf{V}^\star := \mathsf{Eval}^{\mathsf{out}}(g_{T^\star}, (\mathbf{V}_{\ell,k})_{1\le\ell\le\nu,1\le k\le r})$
            $K_0^\star := \mathsf{Round}(\mathbf{SV}^\star\mathbf{t})$
            $\mathbf{U}^\star := \mathsf{Eval}^{\mathsf{in}}(g_{T^\star}, (\tau_\ell[k], \mathbf{U}_{\ell,k})_{1\le\ell\le\nu,1\le k\le r})$
            $K_0^\star := \mathsf{Round}((\mathbf{SAU}^\star + \mathbf{SG})\mathbf{t})$
            $K_1^\star \xleftarrow{\$} \mathcal{K}$
            return $K_b^\star$
    return $\bot$
```

Figure 17: Hybrids for the security proof of the TNIKE from Figure 16. The partial maps pks and sks are initially totally undefined.

Proof. In G_3, the matrix $\widetilde{\mathbf{E}}$ is not sampled directly from $\mathcal{C}^{\mu\times m}_{\tilde{\sigma},p}$. Instead, we sample $\mathbf{E}_1 \leftarrow \mathcal{C}^{\mu\times n}_{\sigma_1,p}$ and $\mathbf{E}_2 \leftarrow \mathcal{C}^{\mu\times m}_{\Sigma,p}$ and set $\widetilde{\mathbf{E}} := \mathbf{E}_1\mathbf{Z} + \mathbf{E}_2$. The matrix $\mathbf{Z}$, which is sampled according to $\mathcal{D}_{\mathbb{Z}_p^{n\times m},\gamma}$, has spectral norm $\sigma_{\mathbf{Z}} \leq C\gamma\sqrt{m}$ with probability at least $1 - 2^{-m}$ by Lemma 2.12. And in this case the two ways to sample $\widetilde{\mathbf{E}}$ yield the the same distribution by Lemma 2.13 using parameter constraint 5. □

Lemma 5.21 ($\mathsf{G}_3 \rightsquigarrow \mathsf{G}_4$).

$$\Pr[\mathsf{G}_3^{\mathcal{A}}(\lambda) \Rightarrow 1] = \Pr[\mathsf{G}_4^{\mathcal{A}}(\lambda) \Rightarrow 1]$$

Proof. In G_4 we change the computation of $K_0^\star$ conceptually. By the correctness of $\mathsf{Eval}^{\mathsf{in}}/\mathsf{Eval}^{\mathsf{out}}$, the games are identical. □

Lemma 5.22 (G_4).

$$\Pr[\mathsf{G}_4^{\mathcal{A}}(\lambda) \Rightarrow 1] \leq \nu\frac{\sqrt{2}\mu}{2}\sqrt{\frac{p^{\ell+1}\nu}{\sigma_1^n}}$$

Proof. We use the GLOHL to argue that in G_4 $K_0^\star$ is statistically close to a uniformly random value.

Notice that, for every user id, his public key $\mathsf{pk} = (\mathbf{SA} + \mathbf{E}, \_)$ can be simulated with the values $\mathbf{SB}$ and $\mathbf{S} + \mathbf{E}_1$ instead of $\mathbf{S}$ and $\mathbf{E}_1$. To do this, we compute $\mathsf{pk}_{\mathsf{I}} := \mathbf{SBC} + (\mathbf{S} + \mathbf{E}_1)\mathbf{Z} + \mathbf{E}_2 + \mathbf{E}'$ and sample $\mathbf{E}'$ from $\mathcal{D}_{\mathbb{Z}_p^{\mu\times m}-(\mathbf{S}+\mathbf{E}_1)\mathbf{Z}-\mathbf{E}_2,\sigma'}$. Since $\mathbb{Z}_p^{\mu\times m} - (\mathbf{S} + \mathbf{E}_1)\mathbf{Z} - \mathbf{E}_2 = \mathbb{Z}_p^{\mu\times m} - \mathbf{E}_1\mathbf{Z} - \mathbf{E}_2 = \mathbb{Z}_p^{\mu\times m} - \widetilde{\mathbf{E}}$, the distribution of $\mathbf{E}'$ stays the same.

The value $\mathbf{SB}$ can leak at most $\ell \log p$ bits min-entropy of each row of $\mathbf{S}$. The value $\mathbf{S} + \mathbf{E}_1$ leaks at most $n\log(p/\sigma_1) + 1$ bits of min-entropy about $\mathbf{S}$ by Lemma 2.19. Thus for the k-th row $\mathbf{S}_i$ of $\mathbf{S}$ we have

$$\widetilde{H}_\infty(\mathbf{S}[k] \mid \mathbf{SB}, \mathbf{S} + \mathbf{E}_1) \geq n\log p - n\log(p/\sigma_1) - 1 - \ell\log p = n\log\sigma_1 - \ell\log p - 1.$$

Let $\mathbf{T}$ be the matrix whose column vectors are the vectors $\mathbf{t}$ contained in the public key and secret key of each user id' where the adversary issues a test query on id and id' and $\mathsf{id} > \mathsf{id}'$. The matrix $\mathbf{GT}$ is then uniformly random (and independent of $\mathbf{S}$) due to the structure of gadget matrix $\mathbf{G}$ and the distribution of the vectors $\mathbf{t}$. And since matrix multiplication is a family of universal hash functions we can use Lemma 2.18 to argue that $\mathbf{S}[k]\mathbf{GT}$ has at most statistical distance

$$\frac{1}{2}\sqrt{2^{-n\log\sigma_1+\ell\log p+1}p\nu} = \frac{\sqrt{2}}{2}\sqrt{\frac{p^{\ell+1}\nu}{\sigma_1^n}}$$

to a uniformly random row vector. Since round outputs a uniformly random bit when the input is uniformly random and the above argument holds in particular for $i = j$, each bit of $K_0^\star$ is statistical close to uniform. □

Combining Lemmata 5.18–5.22 shows Theorem 5.17. □

References

[1] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. “Tightly-Secure Authenticated Key Exchange”. In: *TCC 2015, Part I.* ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. LNCS. Springer, Heidelberg, Mar. 2015, pp. 629–658. DOI: 10.1007/978-3-662-46494-6_26.

[2] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. “On the Impossibility of Tight Cryptographic Reductions”. In: *EUROCRYPT 2016, Part II.* ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 273–304. DOI: 10.1007/978-3-662-49896-5_10.

[3] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. “Possibility and Impossibility Results for Encryption and Commitment Secure under Selective Opening”. In: *EUROCRYPT 2009.* Ed. by Antoine Joux. Vol. 5479. LNCS. Springer, Heidelberg, Apr. 2009, pp. 1–35. DOI: 10.1007/978-3-642-01001-9_1.

[4] Dan Boneh and Ramarathnam Venkatesan. “Breaking RSA May Not Be Equivalent to Factoring”. In: *EUROCRYPT’98.* Ed. by Kaisa Nyberg. Vol. 1403. LNCS. Springer, Heidelberg, 1998, pp. 59–71. DOI: 10.1007/BFb0054117.

[5] Dan Boneh and Mark Zhandry. “Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation”. In: *CRYPTO 2014, Part I.* ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 480–499. DOI: 10.1007/978-3-662-44371-2_27.

[6] Colin Boyd, Wenbo Mao, and Kenneth G. Paterson. “Key Agreement Using Statically Keyed Authenticators”. In: *ACNS 04*. Ed. by Markus Jakobsson, Moti Yung, and Jianying Zhou. Vol. 3089. LNCS. Springer, Heidelberg, June 2004, pp. 248–262. DOI: 10.1007/978-3-540-24852-1_18.

[7] Xavier Boyen, Qixiang Mei, and Brent Waters. “Direct Chosen Ciphertext Security from Identity-Based Techniques”. In: *ACM CCS 2005*. Ed. by Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels. ACM Press, Nov. 2005, pp. 320–329. DOI: 10.1145/1102120.1102162.

[8] Zvika Brakerski and Nico Döttling. “Hardness of LWE on General Entropic Distributions”. In: *EUROCRYPT 2020, Part II*. ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, Heidelberg, May 2020, pp. 551–575. DOI: 10.1007/978-3-030-45724-2_19.

[9] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. “Classical hardness of learning with errors”. In: *45th ACM STOC*. ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 575–584. DOI: 10.1145/2488608.2488680.

[10] Cagatay Capar, Dennis Goeckel, Kenneth G. Paterson, Elizabeth A. Quaglia, Don Towsley, and Murtaza Zafer. “Signal-flow-based analysis of wireless security protocols”. In: *Inf. Comput.* 226 (2013), pp. 37–56. DOI: 10.1016/j.ic.2013.03.004.

[11] David Cash, Eike Kiltz, and Victor Shoup. “The Twin Diffie-Hellman Problem and Applications”. In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 127–145. DOI: 10.1007/978-3-540-78967-3_8.

[12] Jean-Sébastien Coron. “On the Exact Security of Full Domain Hash”. In: *CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. LNCS. Springer, Heidelberg, Aug. 2000, pp. 229–235. DOI: 10.1007/3-540-44598-6_14.

[13] Jean-Sébastien Coron. “Optimal Security Proofs for PSS and Other Signature Schemes”. In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, 2002, pp. 272–287. DOI: 10.1007/3-540-46035-7_18.

[14] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, abhi shelat, and Vinod Vaikuntanathan. “Bounded CCA2-Secure Encryption”. In: *ASIACRYPT 2007*. Ed. by Kaoru Kurosawa. Vol. 4833. LNCS. Springer, Heidelberg, Dec. 2007, pp. 502–518. DOI: 10.1007/978-3-540-76900-2_31.

[15] Ronald Cramer and Victor Shoup. “Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption”. In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, 2002, pp. 45–64. DOI: 10.1007/3-540-46035-7_4.

[16] Whitfield Diffie and Martin E. Hellman. “New Directions in Cryptography”. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[17] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. “Composability and On-Line Deniability of Authentication”. In: *TCC 2009*. Ed. by Omer Reingold. Vol. 5444. LNCS. Springer, Heidelberg, Mar. 2009, pp. 146–162. DOI: 10.1007/978-3-642-00457-5_10.

[18] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”. In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 523–540. DOI: 10.1007/978-3-540-24676-3_31.

[19] Danny Dolev, Cynthia Dwork, and Moni Naor. “Nonmalleable Cryptography”. In: *SIAM Journal on Computing* 30.2 (2000), pp. 391–437.

[20] Régis Dupont and Andreas Enge. “Provably secure non-interactive key distribution based on pairings”. In: *Discrete Applied Mathematics* 154.2 (2006), pp. 270–276.

[21] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. “An Algebraic Framework for Diffie-Hellman Assumptions”. In: *CRYPTO 2013, Part II*. ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 129–147. DOI: 10.1007/978-3-642-40084-1_8.

[22] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. “Non-Interactive Key Exchange”. In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, 2013, pp. 254–271. DOI: 10.1007/978-3-642-36362-7_17.

[23] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. “Tightly CCA-Secure Encryption Without Pairings”. In: *EUROCRYPT 2016, Part I*. ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. LNCS. Springer, Heidelberg, May 2016, pp. 1–27. DOI: 10.1007/978-3-662-49890-3_1.

[24] Craig Gentry, Amit Sahai, and Brent Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: *CRYPTO 2013, Part I*. ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5.

[25] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. “Leveled Fully Homomorphic Signatures from Standard Lattices”. In: *47th ACM STOC*. ed. by Rocco A. Servedio and Ronitt Rubinfeld. ACM Press, June 2015, pp. 469–477. DOI: 10.1145/2746539.2746576.

[26] Jens Groth and Amit Sahai. “Efficient Non-interactive Proof Systems for Bilinear Groups”. In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 415–432. DOI: 10.1007/978-3-540-78967-3_24.

[27] Siyao Guo, Pritish Kamath, Alon Rosen, and Katerina Sotiraki. “Limits on the Efficiency of (Ring) LWE Based Non-interactive Key Exchange”. In: *PKC 2020, Part I*. ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12110. LNCS. Springer, Heidelberg, May 2020, pp. 374–395. DOI: 10.1007/978-3-030-45374-9_13.

[28] Julia Hesse, Dennis Hofheinz, and Lisa Kohl. “On Tightly Secure Non-Interactive Key Exchange”. In: *CRYPTO 2018, Part II*. ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. LNCS. Springer, Heidelberg, Aug. 2018, pp. 65–94. DOI: 10.1007/978-3-319-96881-0_3.

[29] Dennis Hofheinz and Tibor Jager. “Tightly Secure Signatures and Public-Key Encryption”. In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 590–607. DOI: 10.1007/978-3-642-32009-5_35.

[30] Dennis Hofheinz, Tibor Jager, and Edward Knapp. “Waters Signatures with Optimal Security Reduction”. In: *PKC 2012*. Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Vol. 7293. LNCS. Springer, Heidelberg, May 2012, pp. 66–83. DOI: 10.1007/978-3-642-30057-8_5.

[31] Dennis Hofheinz, Eike Kiltz, and Victor Shoup. “Practical Chosen Ciphertext Secure Encryption from Factoring”. In: *Journal of Cryptology* 26.1 (Jan. 2013), pp. 102–118. DOI: 10.1007/s00145-011-9115-0.

[32] Eike Kiltz. “Chosen-Ciphertext Security from Tag-Based Encryption”. In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Heidelberg, Mar. 2006, pp. 581–600. DOI: 10.1007/11681878_30.

[33] Roman Langrehr and Jiaxin Pan. “Tightly Secure Hierarchical Identity-Based Encryption”. In: *Journal of Cryptology* 33.4 (Oct. 2020), pp. 1787–1821. ISSN: 1432-1378. DOI: 10.1007/s00145-020-09356-x.

[34] Allison B. Lewko and Brent Waters. “Why Proving HIBE Systems Secure Is Difficult”. In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 58–76. DOI: 10.1007/978-3-642-55220-5_4.

[35] Vadim Lyubashevsky. “Lattice Signatures without Trapdoors”. In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 738–755. DOI: 10.1007/978-3-642-29011-4_43.

[36] Daniele Micciancio and Chris Peikert. “Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller”. In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41.

[37] Daniele Micciancio and Oded Regev. “Worst-Case to Average-Case Reductions Based on Gaussian Measures”. In: *SIAM J. Comput.* 37.1 (Apr. 2007), 267–302. ISSN: 0097-5397. DOI: 10.1137/S0097539705447360.

[38] Moni Naor, Omer Reingold, and Alon Rosen. “Pseudo-random functions and factoring (extended abstract)”. In: *32nd ACM STOC*. ACM Press, May 2000, pp. 11–20. DOI: 10.1145/335305.335307.

[39] Kenneth G. Paterson and Sriramkrishnan Srinivasan. “Building Key-Private Public-Key Encryption Schemes”. In: *ACISP 09*. Ed. by Colin Boyd and Juan Manuel González Nieto. Vol. 5594. LNCS. Springer, Heidelberg, July 2009, pp. 276–292.

[40] Chris Peikert. “An Efficient and Parallel Gaussian Sampler for Lattices”. In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 80–97. DOI: 10.1007/978-3-642-14623-7_5.

[41] Chris Peikert. “Public-key cryptosystems from the worst-case shortest vector problem: extended abstract”. In: *41st ACM STOC*. ed. by Michael Mitzenmacher. ACM Press, 2009, pp. 333–342. DOI: 10.1145/1536414.1536461.

[42] Chris Peikert and Brent Waters. “Lossy trapdoor functions and their applications”. In: *40th ACM STOC*. ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 187–196. DOI: 10.1145/1374376.1374406.

[43] David Pointcheval and Olivier Sanders. “Forward Secure Non-Interactive Key Exchange”. In: *SCN 14*. Ed. by Michel Abdalla and Roberto De Prisco. Vol. 8642. LNCS. Springer, Heidelberg, Sept. 2014, pp. 21–39. DOI: 10.1007/978-3-319-10879-7_2.

[44] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *37th ACM STOC*. ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[45] Oded Regev. “Quantum Computation and Lattice Problems”. In: *43rd FOCS*. IEEE Computer Society Press, Nov. 2002, pp. 520–529. DOI: 10.1109/SFCS.2002.1181976.

[46] Amit Sahai and Brent Waters. “How to use indistinguishability obfuscation: deniable encryption, and more”. In: *46th ACM STOC*. ed. by David B. Shmoys. ACM Press, 2014, pp. 475–484. DOI: 10.1145/2591796.2591825.

[47] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. “Cryptosystems based on Pairing”. In: *SCIS 2000*. Okinawa, Japan, Jan. 2000.

[48] Brent Waters. “Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions”. In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 619–636. DOI: 10.1007/978-3-642-03356-8_36.

A More Examples for IP-NIKEs

Note that for the following two examples of group-based NIKEs, we use the multiplicative instead of additive notation for groups (e.g., instead of aP for generator P of $\mathbb{G}$, we write g^a for generator g of $\mathbb{G}$).

A.1 Diffie-Hellman NIKE

The Diffie-Hellman key exchange is a NIKE with algorithms $(\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{SharedKey})$ defined as follows. $\mathtt{Setup}(\lambda)$ samples a group generator g of order p for a λ-bit prime p and sets $\mathsf{pp} \leftarrow (g, p)$. $\mathtt{KeyGen}$ gets (g, p), draws $a \leftarrow \mathbb{Z}_p$ uniformly at random and sets $\mathsf{pk} \leftarrow g^a$, $\mathsf{sk} \leftarrow a$. $\mathtt{SharedKey}$ takes pk, sk as input and outputs $\mathsf{pk}^{\mathsf{sk}}$. To see that this gives us a 1-dimensional ip-NIKE, we first define $\mathtt{Ver}, \mathtt{Extract}$ and f_{pp} as follows.

$$
\begin{aligned}
\mathtt{Ver}((g,p), \mathsf{pk}, \mathsf{sk}) &:= 1 \text{ iff } \mathsf{pk} = g^{\mathsf{sk}} \\
\mathtt{Extract}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) &:= (\mathsf{sk}, \mathsf{sk}) \\
\mathtt{PExtract}((g,p), \mathsf{pk}) &:= \log_g(\mathsf{pk}) \\
f((g,p), z) &:= g^z
\end{aligned}
$$

Obviously, the DH NIKE has verifiable keys since $\mathtt{Ver}$ checks keys the same way $\mathtt{KeyGen}$ computes them. Further, the DH NIKE has IP-computable shared keys since $f((g,p), \langle \mathsf{sk}, \mathsf{sk}' \rangle = g^{\langle \mathsf{sk}, \mathsf{sk}' \rangle}) = g^{\mathsf{sk} \cdot \mathsf{sk}'} = \mathsf{pk}^{\mathsf{sk}'} = \mathtt{SharedKey}(\mathsf{pk}, \mathsf{sk}')$, where the second to last equality holds because $\mathtt{Ver}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) = 1$. To prove the binding public keys property, assume a valid key pair $(\mathsf{pk}, \mathsf{sk})$ such that $\mathtt{PExtract}(\mathsf{pp}, \mathsf{pk}) \neq \mathtt{Extract}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})_1$ (where the subscript denotes the first value in the output of $\mathtt{Extract}$). It follows that $\log_g(\mathsf{pk}) \neq \mathsf{sk} \implies g^{\mathsf{sk}} \neq \mathsf{pk} \implies \mathtt{Ver}(\mathsf{pk}, \mathsf{sk}) = 0$, which is a contradiction.

Since the DH NIKE is known to suffer from a loss quadratic in the number of users, the linear loss that follows from applying our Theorem 4.7 is not very interesting.

A.2 NIKE from Hash-Proof-System

Instantiated with the Diffie-Hellman-based hash proof system [15], the HPS-based NIKE [28] is given as follows. $\mathtt{Setup}(\lambda)$ samples generators g, h for a group of order p with λ-bit prime p and sets $\mathsf{pp} \leftarrow (g, h, p)$. $\mathtt{KeyGen}$ gets (g, h, p), draws $a, b, r \leftarrow \mathbb{Z}_p$ and sets $pk \leftarrow (g^a h^b, g^r, h^r)$ and $\mathsf{sk} \leftarrow (a, b, r)$. $\mathtt{SharedKey}(\mathsf{pk}, \mathsf{sk})$ parses $\mathsf{pk} := (\mathsf{hpk}, X, Y)$ and $\mathsf{sk} := (a, b, r)$ and outputs $\mathsf{hpk}^r X^a Y^b$. We can now set

$$
\begin{aligned}
\mathtt{Ver}((g,h,p), (\mathsf{hpk}, X, Y), (a,b,r)) &:= 1 \text{ iff } \mathsf{hpk} = g^a h^b \wedge X = g^r \wedge Y = h^r \\
\mathtt{Extract}((g,h,p), (\mathsf{hpk}, X, Y), (a,b,r)) &:= ((r, sr, a + sb), (a, b, r)) \text{ where } s := \log_g(h) \\
\mathtt{PExtract}((g,h,p), (\mathsf{hpk}, X, Y)) &:= (\log_g(X), \log_g(Y), \log_g(\mathsf{hpk})) \\
f((g,h,p), z) &:= g^z.
\end{aligned}
$$

$\mathtt{Ver}$ is efficiently computable and checks computation as done by $\mathtt{KeyGen}$, hence keys are verifiable. Shared keys are IP-computable since

$$
f((g,h,p), \langle (r, sr, a+sb), (a', b', r') \rangle) = g^{ra' + srb' + ar' + sbr'} = (g^{a'} h^{b'})^r (g^{r'})^a (h^{r'})^b = \mathtt{SharedKey}(\mathsf{pk}, \mathsf{sk}')
$$

with $\mathsf{pk} = (g^a h^b, g^r, h^r), \mathsf{sk}' = (a', b', r')$. For binding public keys, assume a valid key pair $(\mathsf{pk}, \mathsf{sk}) := ((\mathsf{hpk}, X, Y), (a, b, r))$ such that $\mathtt{PExtract}(\mathsf{pp}, \mathsf{pk}) \neq \mathtt{Extract}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$. It follows that $(\log_g(X), \log_g(Y), \log_g(\mathsf{hpk})) \neq (r, sr, a + sb)$ with $s = \log_g(h)$ which contradicts validity, hence such a key pair cannot exist and public keys are binding. Altogether, the HPS-based NIKE instantiated with DDH is a 3-dimensional ip-NIKE.

Theorem 4.7 yields an inherent loss of $N/18$ when proving semi-adaptive security of this NIKE, which is close to the inherent loss of $N/2$ for static (“UF-CKS-heavy“) security proven in [28]. Hence, also for this NIKE our theorem does not yield any new insights.