On Tightly Secure Non-Interactive Key Exchange

Julia Hesse*,1, Dennis Hofheinz†,2, and Lisa Kohl‡,2

1Technische Universität Darmstadt, Germany
julia.hesse@crisp-da.de
2Karlsruhe Institute of Technology, Germany
{dennis.hofheinz,lisa.kohl}@kit.edu

June 11, 2018

Abstract

We consider the reduction loss of security reductions for non-interactive key exchange (NIKE) schemes. Currently, no tightly secure NIKE schemes exist, and in fact Bader et al. (EUROCRYPT 2016) provide a lower bound (of $\Omega(n^2)$, where n is the number of parties an adversary interacts with) on the reduction loss for a large class of NIKE schemes.

We offer two results: the first NIKE scheme with a reduction loss of $n/2$ that circumvents the lower bound of Bader et al., but is of course still far from tightly secure. Second, we provide a generalization of Bader et al.'s lower bound to a larger class of NIKE schemes (that also covers our NIKE scheme), with an adapted lower bound of $n/2$ on the reduction loss. Hence, in that sense, the reduction for our NIKE scheme is optimal.

1 Introduction

Tight security reductions. A security reduction relates the security of a cryptographic construction to the difficulty to solve some assumed-to-be-hard problem. In other words, to base the security of a scheme S on the hardness of a problem P, one has to show how to solve P given an adversary that successfully attacks S. As one usually considers asymptotic security, both adversary and problem solver are required to have polynomial running time and non-negligible success probability.

Many security reductions now guess where in S to embed problem P. For example, in case of a signature scheme, the security reduction might guess in which generated signature (an instance of) P is embedded. Asymptotically, this is fine, as an S-attacker can only ask for a polynomial number of signatures. But when instantiating the scheme with concrete parameters, this guessing step leads to the following paradox: Considering a number of, say, 2^{30} signature queries (which is realistic when thinking of servers) and a security parameter $\lambda = 100$, the concrete loss in success probability introduced by the reduction would actually be larger than a factor of $2^{\lambda/4}$. When aiming at concrete security guarantees (derived from the hardness of P), one thus has to account for the number of expected signatures at the time of set-up, when choosing keylengths.

This makes so called *tight* security reductions a desirable goal. A security reduction is regarded as tight, if (with comparable running times) the success probability of the problem solver is close to the success probability of the underlying attacker. More precisely, one usually requires the success probabilities to only differ up to a small constant factor (or, for a broader notion of tightness, up to a factor linear in the security parameter). Tight security reductions allow to choose the security parameter for concrete instantiation

*Parts of work done while at École Normale Supérieure, Paris, supported by ERC Project CryptoCloud FP7/2007-2013 Grant Agreement no. 339563

†Supported by ERC Project PREP-CRYPTO (724307), and by DFG grants HO 4534/4-1 and HO 4534/2-2.

‡Supported by ERC Project PREP-CRYPTO (724307), by DFG grant HO 4534/2-2 and by a DAAD scholarship. Parts of work done while visiting Interdisciplinary Center Herzliya.

Reference	$\lvert pk\rvert$	model	sec. loss	assumption	uses
Diffie–Hellman [15]	$1 \times \mathbb{G}$	HKR	n^2	DDH	-
Ours, Sec. 3	$3 \times \mathbb{G}$	HKR	$n/2$	DDH	-
CKS08 [10]	$2 \times \mathbb{G}$	DKR	2	CDH	ROM
FHKP13 [18]	$1 \times \mathbb{Z}_N$	DKR	n^2	factoring	ROM
FHKP13 [18]	$2 \times \mathbb{G} + 1 \times \mathbb{Z}_p$	DKR	n^2	DBDH	asymm. pairing
Ours, Sec. 4	$12 \times \mathbb{G}$	DKR	$n/2$	DLIN	symm. pairing

Figure 1: Comparison of existing NIKE schemes. $|pk|$ denotes the size of the public keys, measured in numbers of group elements and exponents. "DKR" or "HKR" denote the CKS-heavy security notion from [18] with dishonest, resp. honest key registrations. Regarding security loss, n denotes the number of honest parties the adversary interacts with and q is the total number of queries made by the adversary. The losses of the two constructions from [18] stems from applying a generic transformation (from the same paper) to level the security guarantees of all compared schemes. Our construction from Section 3 is instantiated with the HPS of Cramer–Shoup based on DDH presented in Section 3.1. We omit the second scheme from [18] since we focus on non-interactive key registration procedures.

independently of the number of expected instantiations (or, say, generated signatures in case of a signature scheme).

Positive and negative results on tight security. Schemes with tight security reductions could already be constructed for a variety of cryptographic applications (such as public-key encryption [6, 29, 2, 39, 38, 21, 27, 20], identity-based encryption [11, 7, 32, 3, 24], digital signature schemes [35, 39, 38, 28, 1], or zero-knowledge proofs [29, 21]). For public-key encryption schemes, the price to pay for an (almost) tight reduction has been reduced to essentially only one additional group element in ciphertexts [21, 20].

On the other hand, starting with the work of Coron [12], a number of works show that certain types of reductions are inherently non-tight (in the sense that a problem solver derived from a given adversary has a significantly reduced success probability). For instance, [12, 33, 30, 4] prove that any "simple" reduction for a sufficiently "structured" signature scheme must lose a factor of $\Omega(q_{\mathsf{sig}})$, where q_{sig} is the number of adversarial signature queries. (Here, the definitions of "simple" and "structured" vary across these papers.) Similar lower bounds exist also for specific schemes and other primitives [19, 43, 37, 17, 4]. Particularly interesting to our case is the work of Bader et al. [4], which proves lower bounds on the reduction loss of signature, encryption, and non-interactive key exchange schemes in the standard model.

Our focus: non-interactive key exchange. In this work, we investigate tight reductions for non-interactive key exchange (NIKE) schemes in the two-party setting[1]. Intuitively, a NIKE scheme enables any two parties P_i and P_j to compute a common shared key K_{ij} using a public-key infrastructure only, but *without any interaction*. (That is, K_{ij} should be an efficiently computable function of P_i's public and P_j's private key, and we require $K_{ij} = K_{ji}$.) Already the original Diffie-Hellman key exchange [15] forms a NIKE scheme (although one that only satisfies a weak form of security). However, the formal investigation of NIKE schemes started with the work of Cash, Kiltz, and Shoup [10], with a more detailed investigation provided in [18].

While there exist highly secure and efficient NIKE schemes (e.g., [10, 18]), currently there is no NIKE scheme with a tight security reduction to a standard assumption (and in the standard model). We believe that this is no coincidence: as we will detail below, the rich interdependencies among NIKE keys prevent existing techniques to achieve tight security. Also, it might be interesting to note that the already mentioned work of Bader et al. [4] presents a particularly strong (i.e., *quadratic*) lower bound of $\Omega(n^2)$ on the reduction loss of NIKE schemes, where n is the number of parties that the adversary interacts with. While the scheme of [10] is proven only in the random oracle model (such that Bader et al.'s lower does not apply), this lower bound applies to the scheme of [18].

[1]We focus on the two-party setting assuming a public key infrastructure (PKI) since this setting allows for efficient standard-model constructions. Intuitively, stronger settings (multi-party, identity-based with/without setup) appear to require qualitatively stronger tools to give any construction at all, tightly secure or not. However, since any n-party NIKE can be viewed as a 2-party NIKE by fixing n-2 identities, our lower bound trivially generalizes to multi-user NIKE schemes.

Our results. In this work, we provide two contributions. First, we construct an efficient and modular NIKE scheme with a reduction significantly tighter than previous reductions. Concretely, our reduction targets the ℓ-Linear assumption in pairing-friendly groups, and has a loss of $n/2$, where n is the number of users an adversary interacts with. Thus, our scheme is the first to break (or, rather, circumvent) the lower bound of Bader et al. [4]. As a technical tool, we also present a generic transformation that turns any mildly secure NIKE scheme (i.e., secure only against passive adversaries) into a strongly secure one (secure against active adversaries).

Second, we show that our security reduction is optimal, in the sense that we can generalize the result of Bader et al. [4] to our scheme, at the price of a smaller lower bound (of precisely $n/2$). Our generalization follows the high-level ideas of Bader et al. (who in turn follow Coron's work [12]). However, unlike their result, we even consider NIKE schemes and reductions that make nontrivial changes to the public-key infrastructure itself. We believe that our second result points out the inherent difference between the public-key or signature settings (in which we already have tightly secure schemes from standard assumptions), and the NIKE setting (in which a broader range of lower bounds holds, and, to our knowledge, no tight schemes exist).

We note that in line with previous works [4, 26], our negative result does not consider schemes or reductions in the random oracle model.

1.1 Technical overview

In order to describe our results, it will be helpful to first recall existing lower bounds results (and in particular the result of Bader et al. [4]). This way, we will be able to detail how we circumvent these lower bounds, and what other obstacles still block the way to a tight reduction.

A closer look on existing lower bound results. It might be interesting to see why these lower bounds do not contradict any of the constructions mentioned above. All mentioned lower bounds use a "meta-reduction" (cf. [9]) that turns any tight reduction into a successful problem solver (even *without* a given successful adversary). To describe how a meta-reduction works, assume a reduction R that interacts with an adversary $\mathcal{A}$. Assume further that R first solves a number of problem instances for $\mathcal{A}$, and then expects $\mathcal{A}$ to solve a new problem instance. (For instance, in the signature setting, R might first generate many signatures for $\mathcal{A}$ on messages of $\mathcal{A}$'s choice, and then expect $\mathcal{A}$ to forge a signature for a fresh message.) R will then try to solve its own input instance using the fresh solution provided by $\mathcal{A}$.

Now a meta-reduction M runs R, and takes the place of $\mathcal{A}$ in an interaction with R. Intuitively, M will try to feed R with R's own problem solutions, and hope that R can use one of those to solve its own input. Of course, security games generally require the adversary to generate a *fresh* problem solution to avoid trivial attacks. (For instance, the standard security game for signatures [23] requires the adversary to forge a signature for a message that has not been signed before.) Hence, M runs R *twice*: in the first run, M asks R for the solutions to, say, q randomly chosen problem instances $z_1, \dots, z_q$. Then, M rewinds R, asks for solutions to *different* problem instances $\tilde{z}_i$, and submits the previously obtained solution to one z_i as fresh solution.

Of course, R may fail to convert a z_i-solution into a solution to its own input *sometimes* (depending on its reduction loss), and this leaves a "loophole" for R to escape the meta-reduction strategy of M. However, a combinatorial argument of [12] shows that R must have a reduction loss of $\Omega(q_{\mathsf{sig}})$ to use this loophole.

For this strategy of M, it is essential that the reduction R will "accept" a problem solution that it has generated itself. To this end, [12, 33] require unique signatures (i.e., problem solutions), and [30, 4] require re-randomizable signatures (so that any valid signature produced by R can be converted in a random signature by M). However, this property is violated (in a very strong sense) by many of the tightly secure signature schemes mentioned above (e.g., [39, 38, 28, 1]). Specifically, the corresponding (tight) reductions find a way to produce special valid-looking signatures for an adversary that are however useless to solving a problem instance. (Of course, these signatures are not re-randomizable or unique.)

The argument of Bader et al. for NIKE schemes. Bader et al. [4] adapt the above argument to NIKE schemes. To describe their argument, we first recall the NIKE security experiment (according to [10]). A NIKE adversary may request an arbitrary number n of public keys pk_i, and may adaptively corrupt an arbitrary subset of them (in which case the adversary gets the corresponding secret keys sk_i).[2] Finally, the

[2]We omit additional capabilities of the adversary which are not relevant for this overview.

adversary selects two public keys $\mathsf{pk}_{i^*}, \mathsf{pk}_{j^*}$ that have not been corrupted, and then must distinguish between their shared key K_{i^*,j^*}, and an independently random value.[3]

Now assume a reduction R that turns any NIKE adversary into a successful problem solver. This reduction R has to be able to answer adversarial corruption queries, and come up with the corresponding secret keys sk_i. Intuitively, a meta-reduction M can take the role of an adversary, and first obtain some of these keys sk_i from R. Then, M can rewind R, and choose to be challenged on a shared key K_{i^*,j^*} that can be computed from one previously obtained sk_i.

The main difference to the signature case above is that n public keys pk_i give rise to $O(n^2)$ shared keys (or, problem instances/solutions) K_{ij}. In particular, $O(n)$ corruptions enable M to compute $O(n^2)$ shared keys (and thus to potentially solve a quadratic number of shared key challenges). If R turns any of those challenge solutions into a problem solution, then M succeeds. Hence, R must fail with probability $1 - O(1/n^2)$. (Another way to view this is that the reduction's success has to vanish with the failures of the simulation.)

How to circumvent the NIKE lower bound. However, similar to previous works, Bader et al. assume that any secret key (or, more generally, problem solution) output by R can be used to solve corresponding challenges posed by R. This assumption can in fact be violated easily, e.g., by allowing many different secret keys per public key. (That is, a secret key is not uniquely determined by a given public key and, e.g., R may hand out different secret keys upon a corruption query.) Furthermore, different secret keys (for a given public key) may behave differently in the computation of shared keys, and thus may not necessarily be useful in solving a given challenge. Similar ideas are at the core of known techniques for improving tightness, in particular in the context of corruptions [5].

While this first thought allows to circumvent the lower bound of Bader et al., its concrete implementation is not clear at all in the context of NIKE schemes. In particular, there should be many secret keys (with different functionality) for a given public key, but the secret keys obtained through corruptions should still satisfy correctness (in the sense that pk_i and sk_j lead to the same shared key as sk_i and pk_j). (We note that this obstacle is specific to NIKE schemes, and in our opinion the main reason why obtaining tightly secure NIKE schemes appears to be particularly difficult.)

Our scheme. To explain our solution, it might be easiest to first outline our scheme (which, in its basic form, is a variation of the password-authenticated key exchange scheme of [34, 22]). Let L be a language, and assume a hash proof system (HPS) for L with public keys hpk and secret keys hsk. We write $H_{\mathsf{hsk}}(x)$ for hash proof of an L-instance x under key hsk. Then, public and secret keys of our NIKE scheme are of the following form:

$$\mathsf{pk} = (\mathsf{hpk}, x) \qquad\qquad \mathsf{sk} = (\mathsf{hsk}, x, w),$$

where $x \in L$ with witness w, and a HPS keypair $(\mathsf{hpk}, \mathsf{hsk})$ are randomly chosen. Given $\mathsf{pk}_i = (\mathsf{hpk}_i, x_i)$ and $\mathsf{sk}_j = (\mathsf{hsk}_j, x_j, w_j)$, the corresponding NIKE shared key is computed as $K_{ij} = H_{\mathsf{hsk}_j}(x_i) \cdot H_{\mathsf{hsk}_i}(x_j)$, where the hash value $H_{\mathsf{hsk}_i}(x_j)$ is computed from (and uniquely determined by) hpk_i and w_j. We have correctness in the sense $K_{ji} = H_{\mathsf{hsk}_i}(x_j) \cdot H_{\mathsf{hsk}_j}(x_i) = H_{\mathsf{hsk}_j}(x_i) \cdot H_{\mathsf{hsk}_i}(x_j) = K_{ij}$.

Recall that there are many HPS secret keys hsk for any given public key hpk. However, all these secret keys act identically on any $x \in L$. Hence, in order to benefit from the non-uniqueness of hsk, a NIKE reduction will have to switch at least one $x \in L$ in a NIKE public key pk_i to a no-instance $x \notin L$. Let us call such a NIKE public key (with $x \notin L$) "invalid". For an invalid pk_i, no (full) secret key exists. This means that our reduction must hope that no invalid pk_i is ever corrupted. Since a NIKE adversary may corrupt all public keys except for the two selected challenge keys $\mathsf{pk}_{i^*}, \mathsf{pk}_{j^*}$, this means that our reduction may instead fail with probability $1 - 2/n$.

In other words, already with one invalid public key, our reduction has a loss of at least $n/2$. On the bright side, we will present a strategy that uses precisely one invalid public key to leverage a NIKE security reduction (with loss $n/2$). This reduction is of course far from tight, but it has a loss still considerably better than the $O(n^2)$ lower bound by Bader et al., and thus is significantly tighter than previous constructions. In a nutshell, our security proof proceeds in game hops:

[3]Like [4], we consider only one challenge pair of public keys (and not an arbitrary number, like the "m-CKS-heavy" notion of [18].

1. We start with the NIKE security game.
2. We guess one index i^*, and hope that pk_{i^*} is one of the challenge public keys finally selected in the adversary's challenge. (If this is not the case, the reduction fails.) Since there are 2 challenge public keys, this step loses a factor of $n/2$.
3. We choose $x_{i^*} \notin L$. Since we may assume that pk_{i^*} is selected as challenge, this change will not be detectable (assuming L has a hard subset membership problem).
4. Finally, we observe that now, *all* keys K_{i^*j} (for arbitrary j) are randomized by the smoothness of the underlying HPS. In fact, HPS smoothness implies that K_{i^*j} is close to uniform, even given pk_j. In particular, this holds for $j = j^*$ and the final challenge $K_{i^*j^*}$.

Note that while [10] also crucially relies on HPSs, there are significant technical differences. Namely, [10] uses hash proof systems mainly as a tool to implement a "replacement decryption method" that allows to forget parts of the secret key. In other words, they use HPSs exclusively in "proof mode". In contrast, for our basic NIKE scheme we use the HPS only in "randomization mode", i.e. to randomize shared keys.

Instantiations and variants.. Our basic scheme only requires a HPS for a language with hard subset membership problem, and thus can be implemented efficiently from various computational assumptions (such as the DDH [13], ℓ-Linear [31], DCR [13], or QR [13] assumptions). However, this basic scheme satisfies only a relatively mild form of security called "honest key registration" or "HKR" security in [18]. Hence, we also present a general transformation that turns *any* mildly secure NIKE scheme into one that satisfies a stronger form of security (dubbed "dishonest key registration" or "DKR" security in [18]). Our scheme requires a suitable non-interactive zero-knowledge proof system, and, very loosely speaking, adapts the Naor-Yung paradigm [41] to NIKE schemes. We finally give a concrete and optimized instance under the ℓ-MDDH assumption [16] (for any $\ell \geq 2$ in pairing-friendly groups). For details we refer to Section 4.3.

We note that we view our construction as a "first" that demonstrates how to circumvent existing lower bounds for a particularly challenging application. We do not claim superior efficiency of our (fully secure) scheme over existing state-of-the-art NIKE schemes, not even when taking into account the reduction loss in the choice of group sizes. Still, Figure 1 provides an overview over existing NIKE schemes, in particular in comparison to our scheme.

Our new lower bound. Even though it breaks the existing bound of Bader et al. [4], the reduction loss (of $O(n)$) of our scheme might be a bit disappointing. Our second result shows that we can extend the results from [4] to show that the reduction loss (at least for our scheme) is optimal. Specifically, we are able to give new lower bounds on the tightness of NIKE reductions even for schemes with invalid public keys.

In more detail, we show that a weak validity check (on public keys) is sufficient to prove a meaningful lower bound. Namely, we require that validity of a public key (in the sense that two valid public keys admit only one shared key) is verifiable given that public key *and one of its possible secret keys.* Hence, as long as a given public key is not corrupted, its validity may not be efficiently verifiable, and a reduction can hope to substitute it with an invalid key. (Note that this is precisely what happens in the proof of our NIKE scheme.)

On the other hand, this weak validity check allows us to again apply a rewinding argument as in [4]. Namely, as soon as the reduction returns a secret key on an extraction query, we can check whether the given public key was actually valid and in this case use the obtained secret key later to compute the unique shared key. The only case where we fail to do so is if the reduction does not return a valid secret key for a certain public key in all rewinding attempts. But then we can simply abort with high probability, namely in case this public key is part of the extraction queries (which happens with probability $1 - 2/n$). In other words, we prove that the best a reduction can do is to switch one public key to invalid and hope that this public key is not part of the extraction queries. We can thus conclude that a NIKE (such as ours) that admits a non-public validity check still suffers from a security reduction loss of at least $n/2$.

In Section 2 we give definitions of non-interactive key exchange and recall existing game-based security notions, as well as the concept of hash proof systems. Further, we provide the definition of non-interactive zero-knowledge proof of knowledge. In Section 3 we present our construction of a mildly secure NIKE with a security reduction whose tightness significantly improves upon existing NIKEs. We further we show how to concretely instantiate our NIKE based on DDH. In Section 4 we show how to transform a mildly secure

NIKE into a strongly secure one, and how to tweak efficiency of this transformation when using our NIKE construction. In Section 5 we finally prove a new lower bound for a broad class of NIKE schemes including ours.

2 Preliminaries

Notation. Throughout the paper, λ denotes the security parameter. We say that a function is *negligible in* λ if its inverse vanishes asymptotically faster than any polynomial in λ. If a probabilistic algorithm $\mathcal{A}$ has running time polynomial in λ, we say that $\mathcal{A}$ is *probabilistic polynomial time* (PPT). We use $y \leftarrow \mathcal{A}(x)$ to denote that y is assigned the output of $\mathcal{A}$ running on input x, and we write $y \leftarrow \mathcal{A}(x;r)$ to make the randomness r used by a probabilistic algorithm explicit. We use $y \overset{\$}{\leftarrow} X$ to denote sampling from a set X uniformly at random. For $n \in \mathbb{N}$ by $[n]$ we denote the set $\{1,\ldots,n\}$. Let $\varepsilon \in [0,1]$ and $\mathcal{X},\mathcal{Y}$ distributions. To denote that $\mathcal{X}$ and $\mathcal{Y}$ have statistical distance at most ε, we write $\mathcal{X} \equiv_\varepsilon \mathcal{Y}$ and say $\mathcal{X}$ and $\mathcal{Y}$ are *ε-close.*

To represent group elements we use the notation introduced in [16]. Namely, for $a \in \mathbb{Z}_p$, we define $[a] = aP \in \mathbb{G}$ as the *implicit representation* of a in $\mathbb{G}$, where the generator P should be clear from the context. To denote a vector we define $[a,b] := (g^a, g^b)$ accordingly. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n\times m}$ we define $[\mathbf{A}]$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}$:

$$[\mathbf{A}] := \begin{pmatrix} a_{11}P & \ldots & a_{1m}P \\ \\ a_{n1}P & \ldots & a_{nm}P \end{pmatrix} \in \mathbb{G}^{n\times m}$$

Note that from $[a] \in \mathbb{G}$ it is hard to compute the value a if the discrete logarithm assumption holds in $\mathbb{G}$. Obviously, given $[a],[b] \in \mathbb{G}$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[a]\cdot x = [ax] \in \mathbb{G}$ and $[a]+[b] = [a+b] \in \mathbb{G}$.

Definition 2.1 (Group generator). *Let* GGen *be an algorithm that on input* 1^λ *returns a description* $\mathcal{G} = (\mathbb{G}, p, P)$*, where* $\mathbb{G}$ *are additive cyclic groups of order* p *for a* 2λ*-bit prime* p *with group generator* P*. Then* GGen *is called a* group generator*.*

We recall the definitions of the Matrix Decision Diffie-Hellman (MDDH) assumption from [16]. As it is sufficient for our purposes we restrict to matrix distributions returning matrices of dimensions $(\ell+1)\times\ell$.

Definition 2.2 (Matrix distribution). *Let* $\ell \in \mathbb{N}$ *and* p *be a* 2λ*-bit prime. We call a PPT algorithm* $\mathcal{D}_\ell$ *a* matrix distribution *if it outputs matrices in* $\mathbb{Z}_p^{(\ell+1)\times\ell}$ *of full rank* ℓ*.*

The $\mathcal{D}_\ell$-Matrix Diffie-Hellman problem in $\mathbb{G}$ is to distinguish the between tuples of the form $([\mathbf{A}],[\mathbf{Aw}])$ and $([\mathbf{A}],[\mathbf{u}])$, for a randomly chosen $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_\ell$, $\mathbf{w} \overset{\$}{\leftarrow} \mathbb{Z}_p^\ell$ and $\mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_p^\ell$.

Definition 2.3 ($\mathcal{D}_\ell$-MDDH). *Let* $\mathcal{D}_\ell$ *be a matrix distribution. We say that the* $\mathcal{D}_\ell$-Matrix Diffie-Hellman ($\mathcal{D}_\ell$-MDDH) assumption *holds relative to a prime order group* $\mathbb{G}$*, if for all PPT adversaries* $\mathcal{A}$*,*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{mddh}}_{\mathcal{A},\mathcal{G},\mathbb{G},\mathcal{D}_\ell}(\lambda) := &|\Pr[\mathcal{A}(\mathcal{G},[\mathbf{A}],[\mathbf{Aw}])=1] \\ &- \Pr[\mathcal{A}(\mathcal{G},[\mathbf{A}],[\mathbf{u}])=1]| \leq \mathrm{negl}(\lambda),\end{aligned}$$

where the probabilities are taken over $\mathcal{G} := (\mathbb{G},P,p) \leftarrow$ GGen(1^λ)*,* $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_\ell, \mathbf{w} \overset{\$}{\leftarrow} \mathbb{Z}_p^\ell, \mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_p^{\ell+1}$*.*

Definition 2.4 (ℓ-Linear). *For* $\ell \in \mathbb{N}$ *and the distribution* $\mathcal{D}_\ell$ *returning matrices of the form*

$$\mathbf{A} := \begin{pmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_\ell \\ 1 & \cdots & 1 & 1 \end{pmatrix}$$

for $a_1,\ldots,a_\ell \overset{\$}{\leftarrow} \mathbb{Z}_p$*, we call* $\mathcal{D}_\ell$*-MDDH the* ℓ-Linear assumption (ℓ-LIN).

For $\ell = 1$ the $\mathcal{D}_1$-LIN assumption equals the DDH assumption we define in the following. For $\ell = 2$ the $\mathcal{D}_2$-LIN assumption is also called DLIN (Decisional Linear) assumption.

Definition 2.5 (DDH)**.** *We say that the* decisional Diffie-Hellman (DDH) assumption *holds relative to a prime order group* $\mathbb{G}$*, if for all PPT adversaries* $\mathcal{A}$ *the advantage*

$$\mathrm{Adv}^{\mathsf{ddh}}_{\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathcal{G}, [a], [b], [ab]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [a], [b], [c]]|$$

is negligible in λ*, where the probabilities are taken over* $\mathcal{G} := (\mathbb{G}, p, P) \overset{\$}{\leftarrow} \mathtt{GGen}(1^\lambda)$*,* $a, b, c \overset{\$}{\leftarrow} \mathbb{Z}_p$*.*

The Kernel-Diffie-Hellman assumption $\mathcal{D}_\ell$-KMDH [40] is a natural *computational analogue* of the $\mathcal{D}_\ell$-MDDH Assumption.

Definition 2.6 ($\mathcal{D}_\ell$-Kernel Diffie-Hellman assumption $\mathcal{D}_\ell$-KMDH)**.** *Let* $\mathcal{D}_\ell$ *be a matrix distribution. We say that the* $\mathcal{D}_\ell$*-Kernel Diffie-Hellman (*$\mathcal{D}_\ell$*-KMDH) assumption holds relative to a prime order group* $\mathbb{G}$ *if for all PPT adversaries* $\mathcal{A}$*,*

$$\mathrm{Adv}^{\mathsf{kmdh}}_{\mathcal{G},\mathbb{G},\mathcal{D}_\ell,\mathcal{A}}(\lambda) := \Pr[\mathbf{c}^\top \mathbf{A} = \mathbf{0} \wedge \mathbf{c} \neq \mathbf{0} \mid [\mathbf{c}] \overset{\$}{\leftarrow} \mathcal{A}(\mathcal{G}, [\mathbf{A}])]$$

is negligible in λ*, where the probabilities are taken over* $\mathcal{G} := (\mathbb{G}, p, P) \leftarrow \mathtt{GGen}(1^\lambda)$*, and* $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_\ell$*.*

The $\mathcal{D}_\ell$-KMDH assumption is a relaxation of the $\mathcal{D}_\ell$-MDDH assumption, a non-zero vector in the kernel of $\mathbf{A}$ can be employed to test membership in the column space of $\mathbf{A}$. This observation is captured in the following lemma from [40].

Lemma 2.7. *For any* $\ell \in \mathbb{N}$ *and any matrix distribution* $\mathcal{D}_\ell$*, the* $\mathcal{D}_\ell$*-MDDH assumption implies the* $\mathcal{D}_\ell$*-KMDH assumption.*

Definition 2.8 (Bilinear group generator)**.** *Let* $\mathtt{GGen}^2$ *be an algorithm that on input* 1^λ *returns a description* $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, P, g_T, e)$*, where* $\mathbb{G}$ *is an additive cyclic group of order* p *for a* 2λ*-bit prime* p *with group generators* P *and* $\mathbb{G}_T$ *is a multiplicatively written cyclic group with order* p *and generator* g_T*. Let further* $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ *be a non-degenerate mapping (that is* $e(P, P) \neq 1$ *) satisfying*

$$e([a], P) = e(P, [a]) = e(P, P)^a.$$

Then $\mathtt{GGen}^2$ *is called a* bilinear group generator*.*

For $a \in \mathbb{Z}_p$ we use $[a]_T$ to denote denote elements g_T^a.

A hash function generator is a probabilistic polynomial time algorithm $\mathcal{H}$ that, on input 1^λ, outputs an efficiently computable function $H : \{0,1\}^* \to \{0,1\}^\lambda$, unless domain and co-domain are explicitly specified.

For $k \in \mathbb{N}$ and matrices $\mathbf{A} \in \mathbb{Z}_p^{2k \times k}$ by $\overline{\mathbf{A}} \in \mathbb{Z}_p^{k \times k}$ we denote the upper square matrix of $\mathbf{A}$ and by $\underline{\mathbf{A}} \in \mathbb{Z}_p^{k \times k}$ the lower one. For a vector $\mathbf{a} \in \mathbb{Z}_p^{2k}$ by $\overline{\mathbf{a}} \in \mathbb{Z}_p^k$ we denote the upper k entries of $\mathbf{a}$ and by $\underline{\mathbf{a}} \in \mathbb{Z}_p^k$ the lower ones.

Definition 2.9 (Collision Resistance)**.** *We say that a hash function generator* $\mathcal{H}$ *outputs* collision-resistant functions H*, if for all PPT adversaries* $\mathcal{A}$ *and* $\mathsf{H} \overset{\$}{\leftarrow} \mathcal{H}(1^\lambda)$ *it holds*

$$\mathrm{Adv}^{\mathsf{cr}}_{\mathcal{H},\mathcal{A}}(\lambda) := \Pr\left[x \neq x' \wedge \mathsf{H}(x) = \mathsf{H}(x') \mid (x, x') \leftarrow \mathcal{A}(1^\lambda, \mathsf{H})\right] \leq \mathrm{negl}(\lambda).$$

2.1 Public key encryption

Definition 2.10 (Public key encryption)**.** *We call a tuple of PPT algorithms* $\mathtt{PKE} := (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ *a* public key encryption scheme *if the following holds.*

- $\mathtt{KeyGen}(1^\lambda)$ *returns a key pair* $(\mathsf{ppk}, \mathsf{psk})$*.*
- $\mathtt{Enc}(\mathsf{ppk}, M)$ *returns a ciphertext* C*.*
- $\mathtt{Dec}(\mathsf{psk}, C)$ *returns a message* M *or a special rejection symbol* $\perp$*.*

```
$\mathrm{Exp}^{\mathsf{ind-cpa}}_{\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2),\mathtt{PKE}}(\lambda)$:
  $(\mathsf{ppk}, \mathsf{psk}) \leftarrow \mathtt{PKE.KeyGen}(1^\lambda)$
  $(M_0, M_1, st) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{ppk})$
  $b \xleftarrow{\$} \{0,1\}$
  $C := \mathtt{Enc}(\mathsf{ppk}, M_b)$
  $b^\star \leftarrow \mathcal{A}_2(st, C)$
  if  $b = b^\star$  output  $1$
  else output $0$
```

Figure 2: IND-CPA experiment.

We further require Correctness*, that is for all* $(\mathsf{ppk}, \mathsf{psk})$ *in the range of* $\mathtt{KeyGen}(1^\lambda)$*, for all messages* M *and for all* C *in the range of* $\mathtt{Enc}(\mathsf{pk}, M)$ *we require*

$$\mathtt{Dec}(\mathsf{sk}, C) = 1.$$

Definition 2.11 (IND-CPA). *Let* PKE *be a public key encryption scheme. We say* PKE *is* IND-CPA secure *if for all PPT adversaries* $\mathcal{A}$ *we have that*

$$\mathrm{Adv}^{\mathsf{ind-cpa}}_{\mathcal{A},PKE}(\lambda) := |\Pr[\mathrm{Exp}^{\mathsf{ind-cpa}}_{\mathcal{A},PKE}(\lambda) \Rightarrow 1] - 1/2|$$

is negligible in λ*, where* $\mathrm{Exp}^{\mathsf{ind-cpa}}_{\mathcal{A},PKE}(\lambda)$ *is defined as in Figure 2 and we require* $|M_0| = |M_1|$.

2.2 Hash proof systems

Definition 2.12 (Subset membership problem). *We call* $\mathtt{SMP} := \mathtt{Setup}$ *a* subset membership problem*, if* Setup *is a PPT algorithm with the following properties.*

$\mathtt{Setup}(1^\lambda)$ *outputs a compact (i.e. with length polynomial in* λ*) description* (X, L, R)*, where* $L \subset X$ *are sets and* R *is an efficiently computable relation with*

$$x \in L \iff \exists \text{ witness } w \text{ with } (x, w) \in R.$$

(We say a relation R *is efficiently computable if given a pair* (x, w) *it can be efficiently checked whether* $(x, w) \in R$*.)*

Further we require for all (X, L, R) *in the image of* Setup *that it is possible to efficiently sample elements* x *uniformly at random from* $X \backslash L$ *(written* $x \xleftarrow{\$} X \setminus L$*) and to sample elements* x *uniformly random from* L *together with witness* w *(written* $(x, w) \xleftarrow{\$} R$*).*

Definition 2.13 (Subset membership assumption). *Let* SMP *be a subset membership problem. We say that the* subset membership assumption *holds for* SMP*, if for all PPT algorithms* $\mathcal{A}$ *it holds that*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{smp}}_{\mathcal{A},SMP}(\lambda) :=& |\Pr[\mathcal{A}(1^\lambda, (X, L, R), x) = 1 | (x, w) \xleftarrow{\$} R] \\ & - \Pr[\mathcal{A}(1^\lambda, (X, L, R), x) = 1 | x \xleftarrow{\$} X \setminus L]|\end{aligned}$$

is negligible in λ*, where* $(X, L, R) \xleftarrow{\$} \mathtt{SMP.Setup}(1^\lambda)$.

We will employ the notion of a hash proof system based on [13].

Definition 2.14 (Hash Proof Systems (HPS)). *Let* SMP *be a subset membership problem. We call* $\mathtt{HPS} := \mathtt{Setup}$ *a hash proof system for* SMP*, if it is a PPT algorithm of the following form.*

`Setup`(1^λ) *first samples public parameters* $\mathcal{PP}_{\mathtt{SMP}} := (X, L, R) \leftarrow \mathtt{SMP.Setup}(1^\lambda)$ *for the underlying subset membership problem. Further* `Setup` *chooses sets* $\mathcal{HSK}, \Pi, \mathcal{HPK}$ *such that elements can be efficiently sampled at random from* $\mathcal{HSK}$ *(denoted* $\mathsf{hsk} \xleftarrow{\$} \mathcal{HSK}$*). Further* `Setup` *chooses an efficiently computable map*

$$\alpha : \mathcal{HSK} \longrightarrow \mathcal{HPK},$$

a family of efficiently computable functions

$$\mathcal{H} := \{H_{\mathsf{hsk}} : X \longrightarrow \Pi \mid \mathsf{hsk} \in \mathcal{HSK}\}$$

and an efficiently computable map

$$F : R \times \mathcal{HPK} \longrightarrow \Pi$$

such that for all $\mathsf{hsk} \in \mathcal{HSK}, \mathsf{hpk} \in \mathcal{HPK}$ *with* $\alpha(\mathsf{hsk}) = \mathsf{hpk}$ *and for all* $(x, w) \in R$ *we have*

$$H_{\mathsf{hsk}}(x) = F(x, w, \mathsf{hpk}).$$

Finally, `Setup` *outputs* $\mathcal{PP} := (\mathcal{PP}_{\mathtt{SMP}}, \mathcal{HSK}, \mathcal{H}, \alpha, F)$*, which contains* $\mathcal{PP}_{\mathtt{SMP}}$ *together with the compact (i.e. with length polynomial in* λ*) description of* $\mathcal{HSK}, \mathcal{H}, \alpha$ *and* F*.*

We need a property of a HPS called smoothness, introduced in [13].

Definition 2.15 (Smoothness)**.** *Let* `SMP` *be a subset membership problem and* `HPS` *be a hash proof system for* `SMP`*. We call* `HPS` ε-smooth *if for all* $\mathcal{PP} := ((X, L, R), \mathcal{HSK}, \mathcal{H}, \alpha, F)$ *in the image of* `HPS.Setup`*, the following distributions are* ε*-close:*

$$\left\{ (x, \mathsf{hpk}, H_{\mathsf{hsk}}(x)) \;\middle|\; \begin{array}{c} \mathsf{hsk} \xleftarrow{\$} \mathcal{K} \\ \mathsf{hpk} := \alpha(\mathsf{hsk}) \\ x \xleftarrow{\$} X \setminus L \end{array} \right\} \equiv_\varepsilon \left\{ (x, \mathsf{hpk}, \pi) \;\middle|\; \begin{array}{c} \mathsf{hsk} \xleftarrow{\$} \mathcal{K} \\ \mathsf{hpk} := \alpha(\mathsf{hsk}) \\ x \leftarrow X \setminus L, \pi \xleftarrow{\$} \Pi \end{array} \right\}.$$

(Recall that Π *is the image set of* H_{hsk}*.) In other words, on statements* x *outside the language* L*, the output of the private evaluation algorithms is* ε*-close to uniformly random even under knowledge of the public key. Note though that this statement only holds as long as no image of* H_{hsk} *on input* $x \in X \setminus L$ *is known.*

2.3 Non-interactive key exchange (NIKE)

We formally define the notion of NIKE, following [10],[18] and also adopting most of their notation. A NIKE scheme `NIKE` consists of three algorithms (`Setup`, `KeyGen`, `SharedKey`), an identity space $\mathcal{IDS}$ and a shared key space $\mathcal{K}$ which is the output space of `SharedKey`.

- `Setup`: On input 1^λ, this probabilistic algorithm outputs the system parameters $\mathcal{PP}$.
- `KeyGen`: On input $\mathcal{PP}$ and an ID ID, this probabilistic algorithm outputs a tuple $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{PK} \times \mathcal{SK}$.
- `SharedKey`: On input of the public parameters $\mathcal{PP}$ and two identity, public key pairs $(\mathsf{ID}_1, \mathsf{pk}_1), (\mathsf{ID}_2, \mathsf{sk}_2)$, this deterministic algorithm outputs a shared key $K_{12} \in \mathcal{K}$. We assume that $\mathcal{K}$ contains a failure symbol $\perp$.

We always require `NIKE` to be perfectly correct, meaning that for all corresponding key pairs $(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{sk}_2)$ generated by `KeyGen` it holds

$$\mathtt{SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2) = \mathtt{SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1) \neq \perp$$

Security. We quickly recall the game-based security notion from [10], called the *CKS model*, with its refinements from [18]. The model is defined via adversarial queries to oracles implemented by a challenger $\mathcal{C}$. The challenger $\mathcal{C}$ keeps track of all honest and corrupt registered identities and their keys. We informally describe the oracles provided to the adversary attacking a NIKE `NIKE` below.

Model	q_{regH}	q_{regC}	q_{extr}	q_{revH}	q_{revC}	q_{test}
DKR CKS-light	2	✓	-	-	✓	1
DKR CKS	✓	✓	-	-	✓	✓
DKR CKS-heavy	✓	✓	✓	✓	✓	1
DKR m-CKS-heavy	✓	✓	✓	✓	✓	✓
HKR CKS-light	2	-	-	-	-	1
HKR CKS	✓	-	-	-	-	✓
HKR CKS-heavy	✓	-	✓	✓	-	1
HKR m-CKS-heavy	✓	-	✓	✓	-	✓

Table 1: Types of queries for different security models, taken from [18], where q_x denotes the maximum number of allowed queries of the adversary to oracle $\mathcal{O}_x$. ✓, - and n mean that an adversary is allowed to make arbitrary, zero or n queries of this type, in an arbitrary order.

- $\mathcal{O}_{\mathsf{regH}}$ for registering an honest user. $\mathcal{C}$ generates a key pair using `NIKE.KeyGen` and hands the public key to the adversary.
- $\mathcal{O}_{\mathsf{regC}}$ for registering a corrupt user. The adversary may introduce a public key without providing the corresponding secret key.
- $\mathcal{O}_{\mathsf{extr}}$ for extracting a secret key of an honest user.
- $\mathcal{O}_{\mathsf{revH}}$ for revealing a shared key of an honest pair of users.
- $\mathcal{O}_{\mathsf{revC}}$ for revealing a shared key between a corrupted and an honest user.
- $\mathcal{O}_{\mathsf{test}}$ for obtaining a challenge. $\mathcal{A}$ provides a pair of users it wishes to be challenged upon. $\mathcal{C}$ then flips a coin and replies either with their real shared key or a random one.

First, $\mathcal{C}$ runs $\mathcal{PP} \overset{\$}{\leftarrow} \mathtt{NIKE.Setup}(1^\lambda)$ and gives $\mathcal{PP}$ to $\mathcal{A}$. Then, the adversary may make an arbitrary number of the above queries, in an arbitrary order. Finally, the adversary outputs a bit $\hat{b}$ and wins if $\hat{b} = b$. Note that the adversary may register each ID only once[4].

To obtain different notions of CKS security, the adversary is restricted in the number of its queries. See Table 1 for a complete list. Notions that admit $\mathcal{O}_{\mathsf{regC}}$ and $\mathcal{O}_{\mathsf{revC}}$ queries are said to *allow dishonest key registrations*, dubbed *DKR*. Notions that do not allow such types of queries are called *with honest key registration*, or *HKR* for short.

In this paper, we are interested in *CKS - heavy* secure NIKE schemes. We provide the corresponding security experiment in Figure 3.

Definition 2.16 (HKR- and DKR-CKS-heavy security). *Let `NIKE` be a NIKE. We say `NIKE` is* CKS-heavy secure with honest key registration*, or* HKR-CKS-heavy secure*, if for any PPT adversary $\mathcal{A}$ the advantage*

$$\mathrm{Adv}^{\mathsf{hkr-cks-heavy}}_{\mathcal{A},\mathit{NIKE}}(\lambda) = |\Pr[\mathrm{Exp}^{\mathsf{hkr-cks-heavy}}_{\mathcal{A},\mathit{NIKE}}(\lambda) \Rightarrow 1] - 1/2|$$

is negligible in λ, where $\mathrm{Exp}^{\mathsf{hkr-cks-heavy}}_{\mathcal{A},\mathit{NIKE}}$ is provided in Figure 2.3. Similarly, we say that `NIKE` is CKS-heavy secure with dishonest key registration*, or* DKR-CKS-heavy secure*, if for any PPT adversary $\mathcal{A}$ the advantage*

$$\mathrm{Adv}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A},\mathit{NIKE}}(\lambda) = |\Pr[\mathrm{Exp}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A},\mathit{NIKE}}(\lambda) \Rightarrow 1] - 1/2|$$

is negligible in λ.

2.4 Non-interactive zero knowledge proof of knowledge

The notion of a quasi-adaptive non-interactive zero-knowledge proof was introduced in [8]. The following definition of non-interactive zero-knowledge is an adaptation of [21] with some differences. Note for instance, that we consider computational zero-knowledge instead of perfect zero-knowledge. We will employ such proofs to generically transform a NIKE which is secure in the HKR-CKS-heavy security model to a NIKE which is secure in the DKR-CKS-heavy security model.

[4]In practice, this can be implemented by appending a counter to an identity string.

```
$\mathrm{Exp}^{[\mathsf{hkr}|\mathsf{dkr}]-\mathsf{cks}-\mathsf{heavy}}_{\mathcal{A},\mathtt{NIKE}}(\lambda)$:
  $\mathcal{PP} \xleftarrow{\$} \mathtt{NIKE.Setup}(1^\lambda)$
  $Q_{\mathsf{regH}} := \emptyset$, $Q_{\mathsf{regC}} := \emptyset$, $Q_{\mathsf{extr}} := \emptyset$, $Q_{\mathsf{rev}} := \emptyset$
  $b^\star \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{H}},\, \mathcal{O}_{\mathsf{regC}}(\cdot),\, \mathcal{O}_{\mathsf{revC}}(\cdot,\cdot)}(\mathcal{PP})$
  if $b = b^\star \wedge \mathsf{ID}_1^\star, \mathsf{ID}_2^\star \notin Q_{\mathsf{extr}}$
   $\wedge \{\mathsf{ID}_1^\star, \mathsf{ID}_2^\star\} \notin Q_{\mathsf{rev}}$
    output $1$
  else
    $b' \xleftarrow{\$} \{0,1\}$
    output $b'$

$\mathcal{O}_{\mathsf{regH}}(\mathsf{ID})$:
  if $(\mathsf{ID},\cdot,\cdot) \notin \mathcal{O}_{\mathsf{regC}} \cup Q_{\mathsf{regH}}$
    $(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathtt{NIKE.KeyGen}(\mathcal{PP},\mathsf{ID})$
    $Q_{\mathsf{regH}} := Q_{\mathsf{regH}} \cup \{(\mathsf{ID},\mathsf{pk},\mathsf{sk})\}$
    return $\mathsf{pk}$
  else return $\perp$

$\mathcal{O}_{\mathsf{regC}}(\mathsf{ID},\mathsf{pk})$:
  if $(\mathsf{ID},\cdot,\cdot) \notin \mathcal{O}_{\mathsf{regH}} \cup \mathcal{O}_{\mathsf{regC}}$
    $Q_{\mathsf{regC}} := Q_{\mathsf{regC}} \cup \{(\mathsf{ID},\mathsf{pk},\perp)\}$
  else return $\perp$

$\mathcal{O}_{\mathsf{extr}}(\mathsf{ID})$:
  if $\exists \mathsf{sk}\colon (\mathsf{ID},\mathsf{pk},\mathsf{sk}) \in Q_{\mathsf{regH}}$
    $Q_{\mathsf{extr}} := Q_{\mathsf{extr}} \cup \{\mathsf{ID}\}$
    return $\mathsf{sk}$
  else return $\perp$
```

```
$\mathcal{O}_{\mathsf{revH}}(\mathsf{ID}_1,\mathsf{ID}_2)$:
  if $\exists \mathsf{sk}_1,\mathsf{sk}_2\colon (\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{sk}_1)$,
           $(\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{sk}_2) \in Q_{\mathsf{regH}}$
    $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{ID}_1,\mathsf{ID}_2\}\}$
    return $\mathtt{NIKE.SharedKey}(\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{ID}_2,\mathsf{sk}_2)$
  else return $\perp$

$\mathcal{O}_{\mathsf{revC}}(\mathsf{ID}_1,\mathsf{ID}_2)$:
  if $\exists \mathsf{sk}_1\colon (\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{sk}_1) \in Q_{\mathsf{regH}}$,
           $(\mathsf{ID}_2,\mathsf{pk}_2,\cdot) \in Q_{\mathsf{regC}}$
    $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{ID}_1,\mathsf{ID}_2\}\}$
    return $\mathtt{NIKE.SharedKey}(\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{ID}_1,\mathsf{sk}_1)$
  if $\exists \mathsf{sk}_2\colon (\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{sk}_2) \in Q_{\mathsf{regH}}$,
           $(\mathsf{ID}_1,\mathsf{pk}_1,\cdot) \in Q_{\mathsf{regC}}$
    $Q_{\mathsf{rev}} := Q_{\mathsf{rev}} \cup \{\{\mathsf{ID}_1,\mathsf{ID}_2\}\}$
    return $\mathtt{NIKE.SharedKey}(\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{ID}_2,\mathsf{sk}_2)$
  else return $\perp$

$\mathcal{O}_{\mathsf{test}}(\mathsf{ID}_1^\star,\mathsf{ID}_2^\star)$:
  $b \xleftarrow{\$} \{0,1\}$
  if $\exists \mathsf{sk}_1^\star,\mathsf{sk}_2^\star\colon (\mathsf{ID}_1^\star,\mathsf{pk}_1^\star,\mathsf{sk}_1^\star)$,
           $(\mathsf{ID}_2^\star,\mathsf{pk}_2^\star,\mathsf{sk}_2^\star) \in Q_{\mathsf{regH}}$
    $K_0 = \mathtt{NIKE.SharedKey}(\mathsf{ID}_1^\star,\mathsf{pk}_1^\star,\mathsf{ID}_2^\star,\mathsf{sk}_2^\star)$
    $K_1 \xleftarrow{\$} \mathcal{K}$
    return $K_b$
  else return $\perp$
```

Figure 3: Experiment for HKR and DKR CKS-heavy security of a NIKE scheme NIKE with shared key space $\mathcal{K}$. The highlighted parts only occur in the setting of dishonest key registration. The oracle $\mathcal{O}_{\mathsf{test}}$ may only be queried once. $\mathcal{O}_{\mathsf{H}}$ comprises the oracles $\mathcal{O}_{\mathsf{regH}}, \mathcal{O}_{\mathsf{revH}}, \mathcal{O}_{\mathsf{extr}}$ and $\mathcal{O}_{\mathsf{test}}$. We use $\cdot$ to denote an arbitrary entry of a tuple. I.e., $\mathcal{O}_{\mathsf{regH}} \setminus \{(\mathsf{ID},\cdot,\cdot)\}$ denotes the set $\mathcal{O}_{\mathsf{regH}}$ without any tuple that contains ID in the first position.

```
$\mathrm{Exp}^{\mathsf{uss}}_{\mathcal{A},\mathsf{PS}}(\lambda)$:
  $(X, L, R) \leftarrow$ SMP.Setup$(1^\lambda)$
  $(\mathsf{crs}, \mathsf{trp}) \xleftarrow{\$}$ PS.Setup$(1^\lambda, (X, L, R))$
  $Q_{\mathsf{sim}} := \emptyset$
  $(x^\star, \Pi^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sim}}(\cdot)}(1^\lambda, \mathsf{crs})$
  if PS.Ver$(\mathsf{crs}, x^\star, \Pi^\star) \wedge x^\star \notin L \wedge x^\star \notin Q_{\mathsf{sim}}$
    output 1
  else output 0

$\mathcal{O}_{\mathsf{sim}}(x)$:
  $Q_{\mathsf{sim}} := Q_{\mathsf{sim}} \cup \{x\}$
  $\Pi \leftarrow$ PS.Sim$(\mathsf{crs}, \mathsf{trp}, x)$
  return $\Pi$
```

Figure 4: Experiment for unbounded simulation soundness of a QANIZK.

Definition 2.17 (QANIZK)**.** *Let* ***SMP*** *be a subset membership problem. Let* $(X, L, R) \leftarrow$ ***SMP.Setup***(1^λ). *A* quasi adaptive non-interactive zero-knowledge proof (QANIZK) *for* ***SMP*** *is a tuple of PPT algorithms* ***PS*** $:=$ (***Setup***, ***Gen***, ***Ver***, ***Sim***) *of the following form.*

- ***Setup***$(1^\lambda, (X, L, R))$ *generates a common reference string* crs *and a trapdoor* trp*. We assume* (X, L, R) *to be part of the* crs*.*
- ***Prove***(crs, x, w) *given a word* $x \in L$ *and a witness* w *with* $R(x, w) = 1$*, outputs a proof* Π*.*
- ***Ver***(crs, x, Π) *on input* crs*,* $x \in X$ *and* Π *outputs a verdict* $b \in \{0, 1\}$*.*
- ***Sim***$(\mathsf{crs}, \mathsf{trp}, x)$ *given a* crs *with corresponding trapdoor* trp *and a word* $x \in X$*, outputs a proof* Π*.*

Further we require the following properties to hold.

Perfect completeness: *For all security parameters* λ*, all* (X, L, R) *in the image of* ***SMP.Setup***(1^λ)*, all* $(\mathsf{crs}, \mathsf{trp})$ *in the range of* ***Setup***$(1^\lambda, (X, L, R))$*, all words* $x \in L$*, all witnesses* w *such that* $R(x, w) = 1$ *and all* Π *in the range of* ***Prove***(crs, x, w) *we have*

$$\textbf{\textit{Ver}}(\mathsf{crs}, x, \Pi) = 1.$$

Computational zero-knowledge: *For all security parameters* λ*, all* (X, L, R) *in the range of* ***SMP.Setup***(1^λ)*, all tuples* $(\mathsf{crs}, \mathsf{trp})$ *in the range of* ***Setup***$(1^\lambda, (X, L, R))$*, we have for all PPT adversaries* $\mathcal{A}$ *that*

$$\mathrm{Adv}^{\mathsf{zk}}_{\mathcal{A},\textbf{\textit{PS}}}(\lambda) := \mid \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{prv}}(\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{sim}}(\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1 \mid$$

is negligible in λ*, where both oracles on input* (x, w) *first check whether* $(x, w) \in R$*. If this is the case,* $\mathcal{O}_{\mathsf{prv}}$ *returns* ***Prove***(crs, x, w) *and* $\mathcal{O}_{\mathsf{sim}}$ *returns* ***Sim***$(\mathsf{crs}, \mathsf{trp}, x)$ *(and* $\perp$ *otherwise).*

The definition of unbounded simulation soundness follows [42, 14].

Definition 2.18 (Unbounded simulation soundness)**.** *Let* ***PS*** *be a QANIZK for a subset membership problem* ***SMP****. We say* ***PS*** *satisfies* unbounded simulation soundness*, if for all PPT adversaries* $\mathcal{A}$ *the advantage*

$$\mathrm{Adv}^{\mathsf{uss}}_{\mathcal{A},\textbf{\textit{PS}}}(\lambda) := |\Pr[\mathrm{Exp}^{\mathsf{uss}}_{\mathcal{A},\textbf{\textit{PS}}}(\lambda) \Rightarrow 1]|$$

is negligible in λ*, where* $\mathrm{Exp}^{\mathsf{uss}}_{\mathcal{A},\textbf{\textit{PS}}}(\lambda)$ *is provided in Figure 4.*

The following definition is tailored to our purposes. We require a strong notion of proof of knowledge in the sense that we need to be able to extract a witness while simulating proofs ourselves.

Definition 2.19 (QANIZK Proof of knowledge)**.** *Let* ***PS'*** *be a QANIZK for a subset membership problem* ***SMP****, where* ***SMP.Setup*** *returns tuples* (X, L, R)*. Let* ***Setup*** *denote an algorithm that, on input* $(1^\lambda, (X, L, R))$ *runs* $(\mathsf{crs}, \mathsf{trp}) \xleftarrow{\$}$ ***PS'.Setup***$(1^\lambda, (X, L, R))$ *and outputs* $(\mathsf{crs}, \mathsf{trp}, \mathsf{extr})$ *with an additional trapdoor* extr*. Let* ***Gen*** $:=$ ***PS'.Gen***, ***Prove*** $:=$ ***PS'.Prove***, ***Ver*** $:=$ ***PS'.Ver***, ***Sim*** $:=$ ***PS'.Sim****. Let further* ***Extract*** *be an algorithm that on input* $(\mathsf{crs}, \mathsf{extr}, x, \Pi)$ *returns a witness* w*. We say* ***PS*** $=$ (***Setup***, ***Gen***, ***Prove***, ***Ver***, ***Sim***, ***Extract***) *is a* QANIZK Proof of Knowledge for SMP (QANIZKPoK)*, if for all PPT adversaries* $\mathcal{A}$ *the advantage*

$$\mathrm{Adv}^{\mathsf{extr}}_{\mathcal{A},\textbf{\textit{PS}}}(\lambda) := \Pr[\mathrm{Exp}^{\mathsf{extr}}_{\mathcal{A},\textbf{\textit{PS}}}(\lambda) \Rightarrow 1]$$

is negligible in λ*, where* $\mathrm{Exp}^{\mathsf{extr}}_{\mathcal{A},\textbf{\textit{PS}}}(\lambda)$ *is as defined in Figure 5.*

```
$\mathrm{Exp}^{\mathsf{extr}}_{\mathcal{A},\mathsf{PS}}(\lambda)$:
  $(X, L, R) \leftarrow \mathtt{SMP.Setup}(1^\lambda)$
  $(\mathsf{crs}, \mathsf{trp}, \mathsf{extr}) \xleftarrow{\$} \mathtt{PS.Setup}(1^\lambda, (X, L, R))$
  $Q_{\mathsf{sim}} := \emptyset$
  $(x^\star, \Pi^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sim}}(\cdot), \mathcal{O}_{\mathsf{extract}}(\cdot,\cdot)}(1^\lambda, \mathsf{crs})$
  $w \leftarrow \mathcal{O}_{\mathsf{extract}}(x^\star, \Pi^\star)$
  if $\mathtt{PS.Ver}(x^\star, \Pi^\star) = 1 \wedge (x^\star, w) \notin R$
   $\wedge x^\star \notin Q_{\mathsf{sim}}$
      output 1
  else output 0

$\mathcal{O}_{\mathsf{sim}}(x)$:
  $Q_{\mathsf{sim}} := Q_{\mathsf{sim}} \cup \{x\}$
  $\Pi \leftarrow \mathtt{PS.Sim}(\mathsf{crs}, \mathsf{trp}, x)$
  return $\Pi$

$\mathcal{O}_{\mathsf{extract}}(x, \Pi)$:
  if $x \notin Q_{\mathsf{sim}}$
     $w \leftarrow \mathtt{PS.Extract}(\mathsf{crs}, \mathsf{extr}, x, \Pi)$
     return $w$
  else return $\perp$
```

Figure 5: Experiment for a extraction in the presence of simulated proofs. The adversary tries to set up a pair (x, Π) such that a witness w is not extractable from Π.

Remark 2.20. *Note that extraction in the presence of simulated proofs implies unbounded simulation soundness, as an adversary outputting a tuple (x, Π) with $x \notin L$ and $\mathtt{PS.Ver}(\mathsf{crs}, x, \Pi) = 1$ trivially wins the proof of knowledge experiment.*

3 Our construction

We now present a NIKE scheme that is secure in the HKR setting. Our reduction loses a factor of $q_{\mathsf{regH}}/2$, where q_{regH} is the number of honest users. Our scheme uses a hash proof system and its security relies on the hardness of the underlying subset membership problem as well as the smoothness of the HPS. It is presented in Figure 3.

Let us first elaborate on why our NIKE scheme does not fall under the impossibility result of Bader et al. [4]. To enforce that the output of a successful NIKE attacker can always be used to solve the challenge given to the reduction, Bader et al. require that the NIKE scheme allows only public keys whose corresponding secret keys are uniquely determined. This way, the shared key between two public keys is uniquely determined and will be useful to solve the challenge. Moreover, the uniqueness condition has to be *efficiently checkable given only a public key.* This essentially prevents a reduction from switching public keys to “invalid” public keys that violate the uniqueness condition. Formally, Bader et al. require an efficient algorithm `PKCheck` for testing uniqueness.

Our scheme does not provide such an algorithm, since essentially deciding uniqueness amounts to deciding a subset membership problem that we assume to be hard. This way, our reduction will have a way to indistinguishably switch one of the public keys to “invalid” by drawing it from outside the subgroup. Note that for such an invalid public key there exist no secret key, since secret keys contain a witness for the public key belonging to the subgroup. While this non-existence of a secret key helps us in arguing security, it also introduces an inherent loss in our reduction; namely, our reduction has to abort whenever the adversary wants to see the secret key corresponding to the invalid key, which occurs with probability $2/q_{\mathsf{regH}}$ and thus results in a loss of $q_{\mathsf{regH}}/2$. We now provide a proof of security that meets exactly this loss.

Theorem 3.1. *Let $\mathtt{SMP}$ be a subset membership problem, and let $\mathtt{HPS}$ be a hash proof system for $\mathtt{SMP}$, such that for all $\mathcal{PP} := (\mathcal{PP}_{\mathtt{SMP}}, \mathcal{HSK}, \mathcal{H}, \alpha, F)$ in the range of $\mathtt{HPS.Setup}$ the image Π of F and all $H_{\mathsf{hsk}} \in \mathcal{H}$ is a commutative multiplicative group. If the subset membership assumption holds for $\mathtt{SMP}$ and if $\mathtt{HPS}$ is ε-smooth with ε negligible in λ, then the NIKE scheme $\mathtt{NIKE}$ described in Figure 6 is a perfectly correct, HKR-CKS-heavy secure NIKE. Further, the reduction to $\mathtt{SMP}$ loses a factor of $q_{\mathsf{regH}}/2$, where q_{regH} is the number of queries to $\mathcal{O}_{\mathsf{regH}}$ that $\mathcal{A}$ makes. More formally, if $\mathcal{A}$ is an adversary with running time $t_{\mathcal{A}}$ against the scheme in the HKR-CKS-heavy model, there exists an adversary $\mathcal{B}$ with running time $t_{\mathcal{B}} \approx t_{\mathcal{A}}$ breaking the subset membership problem $\mathtt{SMP}$ such that*

$$\mathrm{Adv}^{\mathsf{hkr-cks-heavy}}_{\mathcal{A},\mathtt{NIKE}}(\lambda) \ \leq \ q_{\mathsf{regH}}/2 \cdot (\mathrm{Adv}^{\mathsf{smp}}_{\mathcal{B},\mathtt{SMP}}(\lambda) + \varepsilon)$$

```
NIKE.Setup(1^λ)
  $(\mathcal{PP}_{\mathtt{SMP}}, \mathcal{HSK}, \mathcal{H}, \alpha, F) \xleftarrow{\$} \mathtt{HPS.Setup}(1^\lambda)$
  $\mathcal{PP} := (\mathcal{PP}_{\mathtt{SMP}}, \mathcal{HSK}, \mathcal{H}, \alpha, F)$
  return $\mathcal{PP}$

NIKE.SharedKey($\mathcal{PP}$, ID_1, pk_1, ID_2, sk_2)
  parse $\mathcal{PP} =: (\mathcal{PP}_{\mathtt{SMP}}, \mathcal{HSK}, \mathcal{H}, \alpha, F)$
  parse $\mathsf{pk}_1 =: (\mathsf{hpk}_1, x_1)$
  parse $\mathsf{sk}_2 =: (\mathsf{hsk}_2, x_2, w_2)$
  $K_{12} := H_{\mathsf{hsk}_2}(x_1) \cdot F(x_2, w_2, \mathsf{hpk}_1)$
  return $K_{12}$

NIKE.KeyGen($\mathcal{PP}$, ID)
  parse $\mathcal{PP} =: (\mathcal{PP}_{\mathtt{SMP}}, \mathcal{HSK}, \mathcal{H}, \alpha, F)$
  parse $PP_{\mathtt{SMP}} =: (X, L, R)$
  $\mathsf{hsk} \xleftarrow{\$} \mathcal{HSK}$
  $\mathsf{hpk} := \alpha(\mathsf{hsk})$
  $(x, w) \xleftarrow{\$} R$
  $\mathsf{pk} := (\mathsf{hpk}, x)$
  $\mathsf{sk} := (\mathsf{hsk}, x, w)$
  return $(\mathsf{pk}, \mathsf{sk})$
```

Figure 6: Our NIKE scheme. Recall that $\mathcal{H} = \{H_{\mathsf{hsk}} : X \to \Pi \mid \mathsf{hsk} \in \mathcal{K}\}$ is a family of functions and $F : R \times \mathcal{HPK} \to \Pi$ a function (where $\mathcal{HPK}$ is the image of α).

Game	$\mathcal{O}_{\mathsf{regH}}$ if $i = i^\star$	$\mathcal{O}_{\mathsf{extr}}(\mathsf{ID}_{i^\star})$	$\mathcal{O}_{\mathsf{revH}}(\{\mathsf{ID}, \mathsf{ID}_{i^\star}\})$	$\mathcal{O}_{\mathsf{test}}(\{\mathsf{ID}, \mathsf{ID}_{i^\star})\}$	**Explanation**
$\mathbf{G}_0$	$(x,w) \xleftarrow{\$} R$	$\mathsf{sk}_{i^\star}$	$\mathsf{sk}_{i^\star}/\mathsf{sk}$	$\mathsf{sk}_{i^\star}/\mathsf{sk}$	$= \mathrm{Exp}^{\mathsf{hkr-cks-heavy}}_{\mathcal{A},\mathtt{NIKE}}$
$\mathbf{G}_1$	$(x,w) \xleftarrow{\$} R$	$\mathsf{sk}_{i^\star}$	sk	sk	perfect correctness
$\mathbf{G}_2$	$(x,w) \xleftarrow{\$} R$	**abort**	sk	sk	$q_{\mathsf{regH}}/2$ loss
$\mathbf{G}_3$	$x \xleftarrow{\$} X \setminus L$	**abort**	sk	sk	$\mathtt{SMP}$ assumption
$\mathbf{G}_4$	$x \xleftarrow{\$} X \setminus L$	**abort**	sk	$K_0 \leftarrow \mathcal{K}$	smoothness $\mathtt{HPS}$

Figure 7: Games $\mathbf{G}_0$ to $\mathbf{G}_4$ we employ to prove the NIKE presented in Figure 3 HKR-CKS-heavy secure. From game $\mathbf{G}_1$ on the index $i^\star \xleftarrow{\$} q_{\mathsf{regH}}$ is chosen ahead of time. By $\mathsf{ID}_{i^\star}$ we denote the $i^\star$-th registered honest user. The oracle $\mathcal{O}_{\mathsf{test}}$ may only be queried once. In Column 4 and 5, we give the secret key employed to compute $\mathtt{NIKE.SharedKey}$. By denoting the input as a set $\{\cdot\}$ we want to indicate that we consider both inputs $\mathsf{pk}, \mathsf{pk}_{i^\star}$ and $\mathsf{pk}_{i^\star}, \mathsf{pk}$. In game $\mathbf{G}_0$ there is thus two possibility secret keys to be employed, depending on the order of the input.

Proof. Perfect correctness. Let the public parameters be $\mathcal{PP} := (\mathcal{PP}_{\mathtt{SMP}}, \mathcal{HSK}, \mathcal{H}, \alpha, F) \xleftarrow{\$} \mathtt{NIKE.Setup}(1^\lambda)$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}_2)$. Let further $\mathsf{pk}_1 =: (\mathsf{hpk}_1, x_1), \mathsf{pk}_2 =: (\mathsf{hpk}_2, x_2)$ and $\mathsf{sk}_1 =: (\mathsf{hsk}_1, x_1, w_1), \mathsf{sk}_2 =: (\mathsf{hsk}_2, x_2, w_2)$. As $\mathtt{HPS}$ is a hash proof system and as $x_1, x_2 \in L$, $\mathsf{hpk}_1 = \alpha(\mathsf{hsk}_1)), \mathsf{hpk}_2 = \alpha(\mathsf{hsk}_2))$ we have

$$H_{\mathsf{hsk}_2}(x_1) = F(x_1, w_1, \mathsf{hpk}_2) \text{ and } H_{\mathsf{hsk}_1}(x_2) = F(x_2, w_2, \mathsf{hpk}_1).$$

This yields

$$K_{12} = H_{\mathsf{hsk}_2}(x_1) \cdot F(x_2, w_2, \mathsf{hpk}_1) = H_{\mathsf{hsk}_1}(x_2) \cdot F(x_1, w_1, \mathsf{hpk}_2) = K_{21}$$

as required.

CKS-heavy security. We prove that the $\mathtt{NIKE}$ meets CKS-heavy security with honest key registration in a number of hybrid games. We provide an overview of the games in Figure 7. By $\Pr[\mathbf{G}_i]$ we denote the probability that $\mathcal{A}$ wins game $\mathbf{G}_i$.

Game $\mathbf{G}_0$: The real experiment. Game $\mathbf{G}_0$ is the HKR-CKS-heavy experiment as presented in Figure 3, where $\mathcal{A}$ plays with a challenger $\mathcal{C}$. We have thus

$$\mathrm{Adv}^{\mathsf{hkr-cks-heavy}}_{\mathcal{A},\mathtt{NIKE}}(\lambda) = |\Pr[\mathbf{G}_0] - 1/2| \,.$$

Game $\mathbf{G}_1$: Guess the challenge. Recall that by q_{regH} we denote the number of $\mathcal{O}_{\mathsf{regH}}$ queries of $\mathcal{A}$. From game $\mathbf{G}_1$ on, an index $i^\star \leftarrow q_{\mathsf{regH}}$ is chosen ahead of time. The final goal will be to switch the $i^\star$-th registered honest user $\mathsf{ID}_{i^\star}$ to invalid and hope it is part of the test query. As a first step, from game $\mathbf{G}_1$ on we will make $\mathsf{sk}_{i^\star}$ redundant for $\mathcal{O}_{\mathsf{revH}}$ and $\mathcal{O}_{\mathsf{test}}$ queries. Namely, if $\mathcal{A}$ asks a query of this form with input $(\mathsf{ID}, \mathsf{ID}_{i^\star})$ (for an arbitrary identity ID) we will compute the shared key employing sk, where $(\mathsf{ID}, \mathsf{pk}, \mathsf{sk}) \in Q_{\mathsf{regH}}$, instead of $\mathsf{sk}_{i^\star}$. By perfect correctness of `NIKE` we have

$$\Pr[\mathbf{G}_1] = \Pr[\mathbf{G}_0].$$

Game $\mathbf{G}_2$: Abort upon wrong guess. We change the winning condition of the game as follows. If $\mathsf{ID}_{i^\star}$ is not included in the test query of $\mathcal{A}$, the experiment returns 1 with probability 1/2 and aborts. Then it holds

$$\begin{aligned}\Pr[\mathbf{G}_2] &= \Pr[\mathbf{G}_1] \cdot 2/q_{\mathsf{regH}} + 1/2 \cdot (1 - 2/q_{\mathsf{regH}}) \\ &= (\Pr[\mathbf{G}_1] - 1/2) \cdot 2/q_{\mathsf{regH}} + 1/2\end{aligned}$$

and thus

$$\Pr[\mathbf{G}_1] - 1/2 = q_{\mathsf{regH}}/2 \cdot (\Pr[\mathbf{G}_2] - 1/2).$$

Game $\mathbf{G}_3$: Remove the secret key. Upon receiving the $i^\star$-th register honest user query, $\mathcal{C}$ deviates from the `NIKE.KeyGen` procedure as follows: instead of drawing $(x_{i^\star}, w_{i^\star}) \xleftarrow{\$} R$, $\mathcal{C}$ draws $x_{i^\star} \xleftarrow{\$} X \setminus L$. Note that this way there is no $w_{i^\star}$ such that $R(x_{i^\star}, w_{i^\star}) = 1$ and thus $\mathcal{C}$ cannot compute a secret key $\mathsf{sk}_{i^\star}$. Instead, $\mathcal{C}$ adds $(\mathsf{ID}_{i^\star}, \mathsf{pk}_{i^\star}, \mathsf{sk}_{i^\star}) := (\mathsf{ID}_{i^\star}, (\mathsf{hpk}_{i^\star}, x_{i^\star}), (\mathsf{hsk}_{i^\star}, \perp))$ to Q_{regH}. A distinguisher between both games can be turned directly into a SMP attacker $\mathcal{B}$ putting his challenge in the place of $x_{i^\star}$. If the challenge was in L, Game $\mathbf{G}_2$ was simulated, else it was Game $\mathbf{G}_3$. Observe that it is crucial here that $\mathcal{C}$ does not make use of $w_{i^\star}$ anymore due to the changes made in Game 1. This yields

$$|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_3]| \leq \mathrm{Adv}^{\mathsf{smp}}_{\mathcal{B},\mathsf{SMP}}(\lambda).$$

Game $\mathbf{G}_4$: Randomize the test query. $\mathcal{C}$ changes the answer to the query $\mathcal{O}_{\mathsf{test}}(\mathsf{ID}_{i^\star}, \mathsf{ID})$[5] by drawing $K_0 \xleftarrow{\$} \mathcal{K}$, where $\mathcal{K} = \Pi$ is the image of the hash functions of the HPS. To analyze the distinguishing advantage, note that in the former game it holds that $K_0 = \texttt{NIKE.SharedKey}(\mathsf{ID}_{i^\star}, \mathsf{pk}_{i^\star}, \mathsf{ID}, \mathsf{sk}) = H_{\mathsf{hsk}}(x_{i^\star}) \cdot F(x, w, \mathsf{hpk}_{i^\star})$ with $(\mathsf{ID}, \mathsf{pk}, \mathsf{sk}) = (\mathsf{ID}, (\mathsf{hpk}, x), (\mathsf{hsk}, w)) \in Q_{\mathsf{regH}}$ and $(\mathsf{ID}_{i^\star}, \mathsf{pk}_{i^\star}, \mathsf{sk}_{i^\star}) = (\mathsf{ID}_{i^\star}, (\mathsf{hpk}_{i^\star}, x_{i^\star}), (\mathsf{hsk}_{i^\star}, \perp)) \in Q_{\mathsf{regH}}$. The two distributions $(x_{i^\star}, \mathsf{hpk}, H_{\mathsf{hsk}}(x_{i^\star})), (x_{i^\star}, \mathsf{hpk}, r \xleftarrow{\$} \Pi)$ are ε-close by the ε-smoothness of the HPS, and thus K_0 was already statistically close to the uniform distribution over Π in Game $\mathbf{G}_3$. We thus have

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_4]| \leq \varepsilon.$$

We now show that the advantage of $\mathcal{A}$ playing the CKS-heavy game is negligible. We repeatedly use a folklore technique - add zero, then apply the triangle inequality - to go through all the above games until Game $\mathbf{G}_4$, for which the winning probability of $\mathcal{A}$ is 1/2 since its view does not depend on the challenge bit.

$$\begin{aligned}\mathrm{Adv}^{\mathsf{hkr-cks-heavy}}_{\mathcal{A},\texttt{NIKE}}(\lambda) &= |\Pr[\mathbf{G}_0] - 1/2| = |\Pr[\mathbf{G}_1] - 1/2| \\ &= q_{\mathsf{regH}}/2 \cdot |\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_3] + \Pr[\mathbf{G}_3] - 1/2| \\ &\leq q_{\mathsf{regH}}/2 \cdot |\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_4] + \Pr[\mathbf{G}_4] - 1/2| + q_{\mathsf{regH}}/2 \cdot \mathrm{Adv}^{\mathsf{smp}}_{\mathcal{B},\mathsf{SMP}}(\lambda) \\ &\leq q_{\mathsf{regH}}/2 \cdot |\Pr[\mathbf{G}_4] - 1/2| + q_{\mathsf{regH}}/2 \cdot (\mathrm{Adv}^{\mathsf{smp}}_{\mathcal{B},\mathsf{SMP}}(\lambda) + \varepsilon) \\ &= q_{\mathsf{regH}}/2 \cdot (\mathrm{Adv}^{\mathsf{smp}}_{\mathcal{B},\mathsf{SMP}}(\lambda) + \varepsilon)\end{aligned}$$

□

[5] Note that, starting with Game $\mathbf{G}_2$, $i^\star$ is always one of the inputs to $\mathcal{O}_{\mathsf{test}}$.

Remark 3.2. *A variant of our NIKE can be obtained if there is a total ordering $<$ on all identities. Then, the shared key of* $\mathsf{ID}_1, \mathsf{ID}_2$ *can be computed as the hash of the statement provided by the* smaller *identity. More formally, we modify* `NIKE.SharedKey` *as follows:*

$$\begin{aligned}\texttt{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2) :&= H_{hsk_2}(x_1) \\ &= F(x_1, w_1, hpk_2) =: \texttt{NIKE.SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1),\end{aligned}$$

where $\mathsf{ID}_1 < \mathsf{ID}_2$*. The only change in the proof of security is that in game* $\mathbf{G}_2$ *the challenger aborts if the guessed* $i^\star$ *is not the* smallest *identity contained in the test query. This yields a reduction loss of* q_{regH}*.*

3.1 Instantiating our construction

Definition 3.3 ($\mathcal{D}_\ell$-MDDH as subset membership problem)**.** *Let* `GGen` *be a group generator. We define the following SMP based on DDH via the following algorithm* `Setup`*: on input* 1^λ *the algorithm* `Setup` *first samples a group* $\mathcal{G} := (\mathbb{G}, p, P) \leftarrow$ `GGen`(1^λ)*. Further,* `Setup` *draws an* $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_\ell$ *at random. It defines the sets as*

$$\begin{aligned}X &:= \mathbb{G}^{\ell+1}, \\ L &:= \langle[\mathbf{A}]\rangle = \{[\mathbf{x}] \in \mathbb{G}^{\ell+1} \mid \exists \mathbf{w} \in \mathbb{Z}_p^\ell \colon [\mathbf{x}] = [\mathbf{A}] \cdot \mathbf{w}\} \text{ and} \\ R &:= \{([\mathbf{x}], \mathbf{w}) \in \mathbb{G}^{\ell+1} \times \mathbb{Z}_p^\ell \mid [\mathbf{x}] = [\mathbf{A}] \cdot \mathbf{w}\}.\end{aligned}$$

`Setup` *finally returns* $(\mathbb{G}, p, P)$ *and* $[\mathbf{A}]$ *as a compact description of* X, L *and* R*. It is easy to see that under* $\mathcal{D}_\ell$*-MDDH the subset membership assumption holds for* `SMP`*. For* $\ell = 1$ *the language* L *is the so-called* Diffie-Hellman language*.*

Definition 3.4 (HPS for $\mathcal{D}_\ell$-MDDH)**.** *Let* `SMP` *the subset membership problem based on* $\mathcal{D}_\ell$*-MDDH as presented in Definition 3.3. Following [13] we present a HPS* `HPS` *for* `SMP`*. Given a group description* $\mathcal{G} := (\mathbb{G}, p, P)$ *and a matrix* $[\mathbf{A}]$*,* `Setup` *defines* $\mathcal{HSK} := \mathbb{Z}_p^{\ell+1}$*,* $\Pi := \mathbb{G}$*,* $\mathcal{HPK} := \mathbb{G}$*. Further* `Setup` *defines*

$$\begin{aligned}\alpha \colon \mathbb{Z}_p^{\ell+1} &\to \mathbb{G}^{1\times\ell}, \mathsf{hsk} && \mapsto \mathsf{hsk}^\top \cdot [\mathbf{A}], \\ H_{\mathsf{hsk}} \colon \mathbb{G}^{\ell+1} &\to \mathbb{G}, \quad [\mathbf{x}] && \mapsto \mathsf{hsk}^\top \cdot [\mathbf{x}] \\ F \colon \mathbb{G}^{\ell+1} \times \mathbb{Z}_p^\ell \times \mathbb{G}^{1\times\ell} &\to \mathbb{G}, \quad ([\mathbf{x}], \mathbf{w}, [\mathsf{hpk}]) && \mapsto [\mathsf{hpk}] \cdot \mathbf{w}.\end{aligned}$$

This defines a `HPS`*, as for all* $\mathsf{hsk} \in \mathbb{Z}_p^{\ell+1}$*,* $[\mathsf{hpk}] = \alpha(\mathsf{hsk})$*,* $[\mathbf{x}] = [\mathbf{A}] \cdot \mathbf{w} \in L$ *we have*

$$\begin{aligned}H_{\mathsf{hsk}}([\mathbf{x}]) = \mathsf{hsk}^\top \cdot [\mathbf{x}] = \mathsf{hsk}^\top \cdot [\mathbf{A}] \cdot \mathbf{w} &= [\mathsf{hpk}] \cdot \mathbf{w} \\ &= F([\mathbf{x}], \mathbf{w}, [\mathsf{hpk}]).\end{aligned}$$

Lemma 3.5. *The HPS presented in Definition 3.4 is* ε*-smooth for* $\varepsilon = 0$*.*

Proof. Let $\mathbf{A}^\perp \in \mathbb{Z}_p^{(\ell+1)\times\ell}$ such that $(\mathbf{A}^\perp)^\top \cdot \mathbf{A} = \mathbf{0}$. Then for any $\mathsf{hsk} \in \mathbb{G}^{\ell+1}, \mathbf{k} \in \mathbb{Z}_p^\ell$ we have

$$\alpha(\mathsf{hsk} + \mathbf{k} \cdot \mathbf{A}^\perp) = \alpha(\mathsf{hsk}) + \mathbf{k} \cdot (\mathbf{A}^\perp)^\top \cdot \mathbf{A} = \alpha(\mathsf{hsk}).$$

Further, note that the distributions $\{\mathsf{hsk} \mid \mathsf{hsk} \leftarrow \mathbb{Z}_p^{\ell+1}\}$ and $\{\mathsf{hsk} + \mathbf{k} \cdot \mathbf{a}^\perp \mid \mathsf{hsk} \leftarrow \mathbb{Z}_p^{\ell+1}, \mathbf{k} \leftarrow \mathbb{Z}_p^\ell\}$ are equivalent. This yields

$$\begin{aligned}&\left\{([\mathbf{x}], [\mathsf{hpk}], H_{\mathsf{hsk}}([\mathbf{x}])) \;\middle|\; \begin{array}{c}\mathsf{hsk} \overset{\$}{\leftarrow} \mathcal{K}, \\ [\mathsf{hpk}] := \alpha(\mathsf{hsk}), [\mathbf{x}] \overset{\$}{\leftarrow} X \setminus L\end{array}\right\} \\ \equiv &\left\{([\mathbf{x}], [\mathsf{hpk}], H_{\mathsf{hsk}+\mathbf{k}\cdot\mathbf{A}^\perp}([\mathbf{x}])) \;\middle|\; \begin{array}{c}\mathsf{hsk} \overset{\$}{\leftarrow} \mathcal{K}, \mathbf{k} \overset{\$}{\leftarrow} \mathbb{Z}_p^\ell, \\ [\mathsf{hpk}] := \alpha(\mathsf{hsk}), [\mathbf{x}] \overset{\$}{\leftarrow} X \setminus L\end{array}\right\} \\ = &\left\{([\mathbf{x}], [\mathsf{hpk}], H_{\mathsf{hsk}}([\mathbf{x}]) + \mathbf{k} \cdot \underbrace{(\mathbf{A}^\perp)^\top \cdot [\mathbf{x}]}_{\neq 0}) \;\middle|\; \begin{array}{c}\mathsf{hsk} \overset{\$}{\leftarrow} \mathcal{K}, \mathbf{k} \overset{\$}{\leftarrow} \mathbb{Z}_p^\ell, \\ [\mathsf{hpk}] := \alpha(\mathsf{hsk}), [\mathbf{x}] \overset{\$}{\leftarrow} X \setminus L\end{array}\right\} \\ \equiv &\left\{([\mathbf{x}], [\mathsf{hpk}], \pi) \;\middle|\; \begin{array}{c}\mathsf{hsk} \overset{\$}{\leftarrow} \mathcal{K}, \pi \overset{\$}{\leftarrow} \mathbb{G}, \\ [\mathsf{hpk}] := \alpha(\mathsf{hsk}), [\mathbf{x}] \overset{\$}{\leftarrow} X \setminus L\end{array}\right\}.\end{aligned}$$

```
$\mathtt{NIKE}_{\mathsf{dkr}}.\mathtt{Setup}(1^\lambda)$
  $\mathcal{PP} \leftarrow \mathtt{NIKE.Setup}(1^\lambda)$
  $\mathcal{PP}_{\mathtt{PS}} \leftarrow \mathtt{PS.Setup}(1^\lambda, (X_{\mathtt{NIKE}}, L_{\mathtt{NIKE}}, R_{\mathtt{NIKE}}))$
  parse $\mathcal{PP}_{\mathtt{PS}} := (\mathsf{crs}, \mathsf{trp}, \mathsf{extr})$
  $\mathcal{PP}_{\mathsf{dkr}} := (\mathcal{PP}, \mathsf{crs})$
  return $\mathcal{PP}_{\mathsf{dkr}}$

$\mathtt{NIKE}_{\mathsf{dkr}}.\mathtt{SharedKey}(\mathcal{PP}_{\mathsf{dkr}}, \mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$
  parse $\mathcal{PP}_{\mathsf{dkr}} =: (\mathcal{PP}, \mathsf{crs})$
  parse $\mathsf{pk}_1 =: (\mathsf{pk}_1', \Pi_1')$
  if $\mathtt{PS.Ver}(\mathsf{crs}, \mathsf{ID}_1, \mathsf{pk}_1', \Pi_1') = 1$
    return $\mathtt{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1', \mathsf{ID}_2, \mathsf{sk}_2)$
  else return $\perp$

$\mathtt{NIKE}_{\mathsf{dkr}}.\mathtt{KeyGen}(\mathcal{PP}_{\mathsf{dkr}}, \mathsf{ID})$
  parse $\mathcal{PP}_{\mathsf{dkr}} =: (\mathcal{PP}, \mathsf{crs})$
  $r \leftarrow \mathcal{R}_{\mathsf{rand}}$
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}; r)$
  $\Pi \leftarrow \mathtt{PS.Prove}(\mathsf{crs}, \mathsf{ID}, \mathsf{pk}, \mathsf{sk}, r)$
  return $((\mathsf{pk}, \Pi), \mathsf{sk})$
```

Figure 8: A generic transformation from HKR-CKS-heavy security to DKR-CKS-heavy security. $(X_{\mathtt{NIKE}}, L_{\mathtt{NIKE}}, R_{\mathtt{NIKE}})$ is defined as in Remark 4.1.

□

Remark 3.6. *As the keyspace of the hash proof system HPS presented in Definition 3.4 is $\mathcal{HSK} := \mathbb{Z}_p^{\ell+1}$, membership in $\mathcal{HSK}$ is efficiently checkable provided p (which is part of the public parameters returned by HPS.Setup).*

4 Security against dishonest key generation

In this section we want to show how to achieve CKS-heavy security for our scheme allowing dishonest key registrations. That is the adversary is allowed to dishonestly register keys and ask for shared keys where one of the public keys is registered dishonestly.

4.1 A generic transformation

We begin by showing how to generically transform a HKR-CKS-heavy secure NIKE into a DKR-CKS-heavy secure NIKE. To this purpose, we first show how to obtain an SMP from a NIKE.

Remark 4.1. *Every NIKE induces a SMP as follows. Let NIKE be a NIKE with public key space $\mathcal{PK}$ and secret key space $\mathcal{SK}$ and randomness space $\mathcal{R}_{\mathsf{rand}}$. Then we define an SMP $\mathtt{SMP}_{\mathtt{NIKE}}$ as follows. On input 1^λ, $\mathtt{SMP}_{\mathtt{NIKE}}.\mathtt{Setup}$ generates $\mathcal{PP} \leftarrow \mathtt{NIKE.Setup}(1^\lambda)$ and sets*

$$\begin{aligned} X_{\mathtt{NIKE}} &:= \mathcal{IDS} \times \mathcal{PK}, \\ L_{\mathtt{NIKE}} &:= \{(\mathsf{ID}, \mathsf{pk}) \in X \mid \exists \mathsf{sk}, r \colon (\mathsf{pk}, \mathsf{sk}) = \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}; r)\} \text{ and} \\ R_{\mathtt{NIKE}} &:= \{(\mathsf{ID}, \mathsf{pk}, \mathsf{sk}, r) \mid (\mathsf{pk}, \mathsf{sk}) = \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}; r)\}. \end{aligned}$$

Theorem 4.2. *If NIKE is a perfectly correct, HKR-CKS-heavy secure NIKE and PS is an QANIZKPoK for the SMP $\mathtt{SMP}_{\mathtt{NIKE}}$, then the $\mathtt{NIKE}_{\mathsf{dkr}}$ presented in Figure 8 with algorithms $\mathtt{NIKE}_{\mathsf{dkr}}.\mathtt{Setup}$, $\mathtt{NIKE}_{\mathsf{dkr}}.\mathtt{KeyGen}$, $\mathtt{NIKE}_{\mathsf{dkr}}.\mathtt{SharedKey}$ is perfectly correct and secure in the DKR-CKS-heavy model. More precisely, if $\mathcal{A}$ is an adversary on $\mathtt{NIKE}_{\mathsf{dkr}}$ with running time $t_{\mathcal{A}}$, there exists adversaries $\mathcal{B}, \mathcal{B}_1, \mathcal{B}_2$ with running times $t_{\mathcal{B}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2} \approx t_{\mathcal{A}}$ such that*

$$\mathrm{Adv}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A},\mathtt{NIKE}_{\mathsf{dkr}}}(\lambda) \leq \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathtt{PS}}(\lambda) + \mathrm{Adv}^{\mathsf{extr}}_{\mathcal{B}_1,\mathtt{PS}}(\lambda) + \mathrm{Adv}^{\mathsf{hkr-cks-heavy}}_{\mathcal{B}_2,\mathtt{NIKE}}(\lambda).$$

Proof. Perfect correctness directly follows from the perfect correctness of the underlying NIKE together with the perfect completeness of PS. To prove security, we proceed via a series of games. By $\Pr[\mathbf{G}_i]$ we denote the probability that $\mathcal{A}$ wins game $\mathbf{G}_i$.

Game	$\mathcal{O}_{\mathsf{regH}}(\mathsf{ID})$	$\mathcal{O}_{\mathsf{regC}}(\mathsf{ID},\mathsf{pk},\Pi)$	$\mathcal{O}_{\mathsf{revC}}(\cdot,\cdot)$	**Explanation**
$\mathbf{G}_0$	$\mathtt{PS.Prove}(\mathsf{ID},\mathsf{pk},\mathsf{sk};r)$	$\perp$	honest	$=\mathrm{Exp}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A},\mathtt{NIKE}}$
$\mathbf{G}_1$	$\mathtt{PS.Sim}(\mathsf{trp},\mathsf{ID},\mathsf{pk})$	$\perp$	honest	$\mathtt{PS}$ comp. ZK
$\mathbf{G}_2$	$\mathtt{PS.Sim}(\mathsf{trp},\mathsf{ID},\mathsf{pk})$	$\mathtt{PS.Extract}(\mathsf{extr},\mathsf{ID},\mathsf{pk},\Pi)$	honest	$\mathtt{PS}$ PoK
$\mathbf{G}_3$	$\mathtt{PS.Sim}(\mathsf{trp},\mathsf{ID},\mathsf{pk})$	$\mathtt{PS.Extract}(\mathsf{extr},\mathsf{ID},\mathsf{pk},\Pi)$	corrupted	$\mathtt{NIKE}$ perf. corr.

Figure 9: Games $\mathbf{G}_0$ to $\mathbf{G}_3$. Column "$\mathcal{O}_{\mathsf{regH}}(\mathsf{ID})$" gives an overview which algorithm is employed for proving that a public key is computed correctly on a honest key registration query. The column "$\mathcal{O}_{\mathsf{regC}}(\mathsf{ID},\mathsf{pk},\Pi)$" indicates the pair $(\mathsf{ID},\mathsf{pk},\cdot)$ registered on a corrupt key query. The column "$\mathcal{O}_{\mathsf{revC}}(\cdot,\cdot)$" indicates whether the secret key of the honest or the corrupted identity is used to compute the shared key. The last column finally gives an explanation on the indistinguishability of games. We assume $\mathtt{PS.Prove}, \mathtt{PS.Sim}, \mathtt{PS.Extract}_2$ to have access to crs without stating it as an input.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is the DKR-CKS-heavy security experiment. We thus have

$$\mathrm{Adv}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A},\mathtt{NIKE}_{\mathsf{dkr}}}(\lambda) = |\Pr[\mathbf{G}_0] - 1/2|.$$

Game $\mathbf{G}_1$: We change the oracle $\mathcal{O}_{\mathsf{regH}}$. Instead of honestly generating a proof $\Pi \leftarrow \mathtt{PS.Prove}(\mathsf{crs},\mathsf{ID},\mathsf{pk},\mathsf{sk};r)$ we simulate $\Pi \leftarrow \mathtt{PS.Sim}(\mathsf{crs},\mathsf{trp},\mathsf{ID},\mathsf{pk})$. By the computational zero-knowledge of $\mathtt{PS}$ there exists an adversary $\mathcal{B}$ with $t_{\mathcal{B}} \approx t_{\mathcal{A}}$ and

$$|\Pr[\mathbf{G}_0] - \Pr[\mathbf{G}_1]| \leq \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathtt{PS}}(\lambda).$$

(The adversary $\mathcal{B}$ obtains $1^\lambda, \mathsf{crs}$, generates $\mathcal{PP} \leftarrow \mathtt{NIKE.Setup}(1^\lambda)$ and forwards $(\mathcal{PP},\mathsf{crs})$ to $\mathcal{A}$. To generate a proof for a public key pk obtained as $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathtt{NIKE.KeyGen}(\mathcal{PP},\mathsf{ID};r)$ it employs its proof generating oracle on input $(\mathsf{ID},\mathsf{pk},\mathsf{sk},r)$. If the oracle was a $\mathtt{PS.Prove}(\cdot,\cdot)$ oracle $\mathcal{B}$ simulates $\mathbf{G}_0$, otherwise $\mathbf{G}_1$. For verifying proofs crs is sufficient.)

Game $\mathbf{G}_2$: We change the behaviour of the oracle $\mathcal{O}_{\mathsf{regC}}$. On input $(\mathsf{ID},\mathsf{pk},\Pi)$, the oracle employs the knowledge extractor $\mathtt{PS.Extract}(\mathsf{crs},\mathsf{extr},\mathsf{ID},\mathsf{pk},\Pi)$ to extract (sk,r) corresponding to $(\mathsf{ID},\mathsf{pk})$ and to register $(\mathsf{ID},\mathsf{pk},\mathsf{sk})$ in Q_{regC}. If

$$\mathsf{bad} := \mathtt{PS.Ver}(\mathsf{crs},\mathsf{ID},\mathsf{pk},\Pi) = 1 \ \wedge \ (\mathsf{ID},\mathsf{pk},\mathsf{sk},r) \notin R_{\mathtt{NIKE}}$$

occurs, we abort. If bad does not occur, $\mathbf{G}_1$ is distributed exactly as $\mathbf{G}_2$, since none of the extracted secret keys is used. On the other hand, an adversary $\mathcal{A}$ playing in $\mathbf{G}_2$ causing bad can be directly turned into an adversary $\mathcal{B}_1$ winning the proof of knowledge experiment, with running time $t_{\mathcal{B}_1} \approx t_{\mathcal{A}}$. Note that bad meets the winning conditions of the PoK experiment, as $(\mathsf{ID},\mathsf{pk}) \notin Q_{\mathsf{sim}}$ is guaranteed by $\mathcal{O}_{\mathsf{regC}}$ rejecting already honestly registered identities. This yields

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \leq \mathrm{Adv}^{\mathsf{extr}}_{\mathcal{B}_1,\mathtt{PS}}(\lambda).$$

Game $\mathbf{G}_3$: We now let $\mathcal{O}_{\mathsf{revC}}$ compute shared keys with the extracted secret keys. Due to the perfect correctness of $\mathtt{NIKE}_{\mathsf{dkr}}$, we have

$$\Pr[\mathbf{G}_2] = \Pr[\mathbf{G}_3].$$

We can now reduce the HKR-CKS-heavy security of $\mathtt{NIKE}$ to the altered DKR-CKS-heavy experiment as described in Game $\mathbf{G}_3$. To this end, we assume an adversary $\mathcal{A}$ winning game $\mathbf{G}_3$ and show how to construct an HKR-CKS-heavy adversary $\mathcal{B}_2$. On input $\mathcal{PP}$, $\mathcal{B}_2$ sets up $(\mathsf{crs},\mathsf{trp},\mathsf{extr}) \leftarrow \mathtt{PS.Extract}(1^\lambda,(X_{\mathtt{NIKE}},L_{\mathtt{NIKE}},R_{\mathtt{NIKE}}))$ and forwards $(\mathcal{PP},\mathsf{crs})$ to $\mathcal{A}$. Further $\mathcal{B}_2$ maintains a set $Q_{\mathsf{regH}} := \emptyset$ and a set $Q_{\mathsf{regC}} := \emptyset$. We next consider oracle queries.

$\mathcal{O}_{\mathsf{extract}}, \mathcal{O}_{\mathsf{revH}}, \mathcal{O}_{\mathsf{test}}$: The adversary $\mathcal{B}_2$ forwards the ID(s) to its own corresponding oracles and hands the answers back to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{regH}}(\mathsf{ID})$**:** On honest register queries, $\mathcal{B}_2$ forwards the ID to its own oracle, on output pk simulates a proof $\Pi \leftarrow \mathtt{PS.Sim}(\mathsf{crs}, \mathsf{trp}, \mathsf{ID}, \mathsf{pk})$, forwards (pk, Π) to $\mathcal{A}$ and sets $Q_{\mathsf{regH}} := Q_{\mathsf{regH}} \setminus \{(\mathsf{ID}, \cdot, \cdot)\}$ and $Q_{\mathsf{regH}} := Q_{\mathsf{regH}} \cup \{(\mathsf{ID}, \mathsf{pk}, \Pi)\}$. (Note that for an identity ID with $(\mathsf{ID}, \cdot, \cdot) \in Q_{\mathsf{regC}}$, i.e., that is already registered as dishonest ID, $\mathcal{B}_2$ forwards $\perp$ to $\mathcal{A}$.)

$\mathcal{O}_{\mathsf{regC}}(\mathsf{ID}, \mathsf{pk}, \Pi)$**:** On corrupt register queries, $\mathcal{B}_2$ checks the proof Π. If it holds $\mathtt{PS.Ver}(\mathsf{crs}, \mathsf{ID}, \mathsf{pk}, \Pi) = 1$, $\mathcal{B}_2$ extracts a secret sk, r corresponding to ID, pk employing its extraction oracle and sets $\mathcal{Q}_{\mathsf{regC}} := \mathcal{Q}_{\mathsf{regC}} \setminus \{(\mathsf{ID}, \cdot, \cdot)\}$ and $\mathcal{Q}_{\mathsf{regC}} := \mathcal{Q}_{\mathsf{regC}} \cup \{(\mathsf{ID}, \mathsf{pk}, \mathsf{sk})\}$. (Note that for an identity ID with $(\mathsf{ID}, \cdot, \cdot) \in Q_{\mathsf{regH}}$, i.e., that is already registered as honest ID, $\mathcal{B}_2$ forwards $\perp$ to $\mathcal{A}$.)

$\mathcal{O}_{\mathsf{revC}}(\mathsf{ID}_1, \mathsf{ID}_2)$**:** For dishonest reveal queries $\mathcal{B}_2$ checks whether exactly one of the keys was registered honestly and the other correct (employing the sets $\mathcal{Q}_{\mathsf{regH}}$ and $\mathcal{Q}_{\mathsf{regC}}$). If this is the case $\mathcal{B}_2$ can employ the extracted secret key to compute the shared key.

Finally, $\mathcal{B}_2$ forwards the output $b^\star$ of $\mathcal{A}$ to its own oracle. It is straightforward to see that $\mathcal{B}_2$ simulates the experiment described in game $\mathbf{G}_3$ perfectly (refer to Figure 9 for an overview of how the game is implemented), and that $\mathcal{B}_2$ wins the HKR-CKS-heavy experiment if $\mathcal{A}$ wins game $\mathbf{G}_3$.

This yields

$$|\Pr[\mathbf{G}_3] - 1/2| \leq \mathrm{Adv}^{\mathsf{hkr-cks-heavy}}_{\mathcal{B}_2, \mathtt{NIKE}}.$$

Altogether, we thus have

$$\begin{aligned}
&\mathrm{Adv}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A}, \mathtt{NIKE}_{\mathsf{dkr}}}(\lambda) = |\Pr[\mathbf{G}_0] - 1/2| \\
&\leq |\Pr[\mathbf{G}_1] - 1/2| + \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B}, \mathtt{PS}}(\lambda) \\
&\leq |\Pr[\mathbf{G}_2] - 1/2| + \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B}, \mathtt{PS}}(\lambda) + \mathrm{Adv}^{\mathsf{extr}}_{\mathcal{B}_1, \mathtt{PS}}(\lambda) \\
&= |\Pr[\mathbf{G}_3] - 1/2| + \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B}, \mathtt{PS}}(\lambda) + \mathrm{Adv}^{\mathsf{extr}}_{\mathcal{B}_1, \mathtt{PS}}(\lambda) \\
&\leq \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B}, \mathtt{PS}}(\lambda) + \mathrm{Adv}^{\mathsf{extr}}_{\mathcal{B}_1, \mathtt{PS}}(\lambda) + \mathrm{Adv}^{\mathsf{hkr-cks-heavy}}_{\mathcal{B}_2, \mathtt{NIKE}}(\lambda)
\end{aligned}$$

□

4.2 Instantiating PS

Let NIKE be a NIKE with public key space $\mathcal{PK}$ and identity space $\mathcal{IDS}$. We can instantiate PS as follows. Let PKE be a CPA-secure encryption scheme with message space $\mathcal{SK} \times \mathcal{R}_{\mathsf{rand}}$ and ciphertext space $\mathcal{C}_{\mathsf{enc}}$. Let $\mathtt{SMP}_{\mathtt{NIKE,PKE}}$ be the SMP that generates $\mathcal{PP} \leftarrow \mathtt{NIKE.Setup}(1^\lambda)$ and $(\mathsf{ppk}, \mathsf{psk}) \leftarrow \mathtt{PKE.KeyGen}(1^\lambda)$ and outputs $(X_{\mathtt{NIKE,PKE}}, L_{\mathtt{NIKE,PKE}}, R_{\mathtt{NIKE,PKE}})$ as

$$X_{\mathtt{NIKE,PKE}} := \mathsf{ID} \times \mathcal{PK} \times \mathcal{C}_{\mathsf{enc}}$$

$$\begin{aligned}
L_{\mathtt{NIKE,PKE}} := \{(\mathsf{ID}, \mathsf{pk}, C) \mid \exists r, \mathsf{sk}, r_{\mathsf{enc}} \colon (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}; r),\\
C = \mathtt{Enc}_{\mathsf{ppk}}(\mathsf{sk}, r; r_{\mathsf{enc}})\}
\end{aligned}$$

$$\begin{aligned}
R_{\mathtt{NIKE,PKE}} := \{(\mathsf{ID}, \mathsf{pk}, \mathsf{sk}, r) \mid (\mathsf{pk}, \mathsf{sk}) = \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}; r),\\
C = \mathtt{Enc}_{\mathsf{ppk}}(\mathsf{sk}, r; r_{\mathsf{enc}})\}
\end{aligned}$$

Let further $\mathtt{PS}'$ be an unbounded simulation-sound QANIZK for $\mathtt{SMP}_{\mathtt{NIKE,PKE}}$ (for an instantiation see [25], for a tight instantiation see [29]). Then the proof system PS provided in Figure 10 is a QANIZK proof of knowledge for $\mathtt{SMP}_{\mathtt{NIKE}}$. More precisely, for every adversary $\mathcal{A}$ with running time $t_{\mathcal{A}}$ we have adversaries $\mathcal{B}, \mathcal{B}_1, \mathcal{B}_2$ with running times $t_{\mathcal{B}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2} \approx t_{\mathcal{A}}$ and $\mathrm{Adv}^{\mathsf{zk}}_{\mathcal{A}, \mathtt{PS}}(\lambda) \leq \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B}, \mathtt{PS}'}(\lambda) + \mathrm{Adv}^{\mathsf{ind-cpa}}_{\mathcal{B}_1, \mathtt{PKE}}(\lambda)$ and $\mathrm{Adv}^{\mathsf{extr}}_{\mathcal{A}, \mathtt{PS}}(\lambda) \leq \mathrm{Adv}^{\mathsf{uss}}_{\mathcal{B}_2, \mathtt{PS}'}(\lambda)$. Note though that instantiating $\mathtt{PS}'$ in general requires a USS-QANIZK for general NP-languages and thus leads to a very inefficient NIKE. We therefore provide a more efficient transformation for our concrete scheme in the following section.

Sketch. **Computational zero-knowledge.** Let $\mathcal{A}$ be an adversary breaking the computational zero-knowledge of PS. We proceed with the proof employing a number of hybrid games. By $\Pr[\mathbf{G}_i]$ we denote the probability that $\mathcal{A}$ wins game $\mathbf{G}_i$.

```
PS.Setup(1^λ, (X_NIKE, L_NIKE, R_NIKE)):
  $(\mathsf{ppk}, \mathsf{psk}) \leftarrow \mathtt{PKE.KeyGen}(1^\lambda)$
  $(\mathsf{crs}', \mathsf{trp}') \leftarrow \mathtt{PS}'.\mathtt{Setup}(1^\lambda, \mathtt{SMP}_{\mathtt{NIKE,PKE}})$
  $\mathsf{crs} := (\mathsf{crs}', \mathsf{ppk})$, $\mathsf{trp} := \mathsf{trp}'$, $\mathsf{extr} := \mathsf{psk}$
  return $(\mathsf{crs}, \mathsf{trp}, \mathsf{extr})$

PS.Extract(crs, extr, pk, Π):
  parse $\mathsf{crs} =: (\mathsf{crs}', \mathsf{ppk})$
  parse $\Pi =: (C, \Pi')$
  if $\mathtt{PS}'.\mathtt{Ver}(\mathsf{crs}', (\mathsf{pk}, C), \Pi') = 0$
    return $\perp$
  $\mathsf{sk} \leftarrow \mathtt{PKE.Dec}(\mathsf{psk}, C)$
  return $\mathsf{sk}$

PS.Prove(crs, pk, sk; r):
  parse $\mathsf{crs} =: (\mathsf{crs}', \mathsf{ppk})$
  draw $r_{\mathsf{enc}}$
  $C \leftarrow \mathtt{Enc}_{\mathsf{ppk}}(\mathsf{sk}, r, r_{\mathsf{enc}})$
  $\Pi' \leftarrow \mathtt{PS}'.\mathtt{Prove}(\mathsf{crs}', (\mathsf{pk}, C), \mathsf{sk}; r, r_{\mathsf{enc}})$
  $\Pi := (C, \Pi')$
  return $\Pi$

PS.Sim(crs, trp, pk):
  parse $\mathsf{crs} =: (\mathsf{crs}', \mathsf{ppk})$
  draw $r_{\mathsf{enc}}$
  $C \leftarrow \mathtt{Enc}_{\mathsf{ppk}}(\mathbf{0}; r_{\mathsf{enc}})$
  $\Pi' \leftarrow \mathtt{PS}'.\mathtt{Sim}(\mathsf{crs}', \mathsf{trp}', (\mathsf{pk}, C))$
  $\Pi := (C, \Pi')$
  return $\Pi$

PS.Ver(crs, pk, Π):
  parse $\mathsf{crs} =: (\mathsf{crs}', \mathsf{ppk})$
  parse $\Pi =: (C, \Pi')$
  $b \leftarrow \mathtt{PS}'.\mathtt{Ver}(\mathsf{crs}', (\mathsf{pk}, C), \Pi')$
  return $b$
```

Figure 10: An unbounded simulation-sound QANIZK PPOK $\mathtt{PS}$ for $\mathtt{SMP}_{\mathtt{NIKE}}$, where $\mathtt{SMP}_{\mathtt{NIKE,PKE}} := (X_{\mathtt{NIKE,PKE}}, L_{\mathtt{NIKE,PKE}}, R_{\mathtt{NIKE,PKE}})$. Here $\mathbf{0}$ is such that $|\mathbf{0}| = |\mathsf{sk}, r|$ for $\mathsf{sk} \in \mathcal{SK}, r \in \mathcal{R}_{\mathsf{rand}}$.

Game $\mathbf{G}_0$: This is the Zero-knowledge game where $\mathcal{A}$ is provided either with an oracle $\mathcal{O}_{\mathsf{prv}}$ or with an oracle $\mathcal{O}_{\mathsf{sim}}$. We have

$$\mathrm{Adv}^{\mathsf{zk}}_{\mathcal{A},\mathtt{PS}}(\lambda) = |\Pr[\mathbf{G}_0] - 1/2|.$$

Game $\mathbf{G}_1$: We switch the proofs Π' in $\mathcal{O}_{\mathsf{prv}}(\cdot,\cdot)$ to simulated. This yields and adversary $\mathcal{B}$ with $t_{\mathcal{B}} \approx t_{\mathcal{A}}$ and

$$|\Pr[\mathbf{G}_0] - \Pr[\mathbf{G}_1]| \leq \mathrm{Adv}^{\mathsf{zk}}_{\mathcal{B}_1,\mathtt{PS}'}(\lambda).$$

Game $\mathbf{G}_2$: We now employ the IND-CPA security of $\mathtt{PKE}$ to switch the ciphertexts in $\mathcal{O}_{\mathsf{prv}}(\cdot,\cdot)$ to encryptions of $\mathbf{0}$. This yields and adversary $\mathcal{B}_1$ with $t_{\mathcal{B}_1} \approx t_{\mathcal{A}}$ and

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \leq \mathrm{Adv}^{\mathsf{ind-cpa}}_{\mathcal{B}_1,\mathtt{PKE}}(\lambda).$$

As now $\mathcal{O}_{\mathsf{prv}}$ and $\mathcal{O}_{\mathsf{sim}}$ behave accordingly we have $\Pr[\mathbf{G}_2] = 1/2$.

Proof of knowledge. Let $\mathcal{A}$ an adversary on the proof of knowledge game. We construct an adversary $\mathcal{B}_2$ on the USS of $\mathtt{PS}'$ as follows. On input $1^\lambda, \mathsf{crs}'$ the adversary $\mathcal{B}_2$ sets up $(\mathsf{ppk}, \mathsf{psk}) \xleftarrow{\$} \mathtt{PKE.KeyGen}(1^\lambda)$ itself and forwards $\mathsf{crs} := (\mathsf{crs}', \mathsf{ppk})$ to $\mathcal{A}$. Simulation queries $(\mathsf{ID}, \mathsf{pk})$ it answers by choosing $r, \mathsf{sk}, r_{\mathsf{enc}}$ and employing its own simulation oracle on $(\mathsf{ID}, \mathsf{pk}, C)$ with $C \leftarrow \mathtt{PKE.Enc}_{\mathsf{ppk}}(\mathsf{sk}, r; r_{\mathsf{enc}})$. It forwards C together with the reply Π' to $\mathcal{A}$. On an extraction query $\mathcal{O}_{\mathsf{extract}}(\mathsf{ID}, \mathsf{pk}, \Pi)$ the adversary parses $\Pi =: (C, \Pi')$ and returns $(\mathsf{sk}, r) \leftarrow \mathtt{PKE.Dec}_{\mathsf{psk}}(C)$ to $\mathcal{A}$. Finally, if $\mathcal{A}$ manages to supply a fresh and valid tuple $(\mathsf{ID}^\star, \mathsf{pk}^\star, \Pi^\star)$ where we are not able to extract a witness, by the correctness of $\mathtt{PKE}$ we have $(\mathsf{ID}^\star, \mathsf{pk}^\star, C^\star) \notin L$ (where $\Pi^\star =: (C^\star, \Pi'^\star)$ and $\mathcal{B}_2$ wins its own experiment by outputting $(ID^\star, \mathsf{pk}^\star, C^\star, \Pi'^\star)$. This yields

$$\mathrm{Adv}^{\mathsf{extr}}_{\mathcal{A},\mathtt{PS}}(\lambda) \leq \mathrm{Adv}^{\mathsf{uss}}_{\mathcal{B}_2,\mathtt{PS}'}(\lambda).$$

□

4.3 An optimized transformation for our construction

Definition 4.3 ($\mathcal{D}_\ell$-MDDH as SMP with pairing)**.** *Let* GGen^2 *be a bilinear group generator. We define the following SMP based on DDH via the following algorithm* $\mathtt{Setup}$*: on input* 1^λ *the algorithm* $\mathtt{Setup}$ *first samples a group* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, p, P, g_T, e) \leftarrow \mathsf{GGen}^2(1^\lambda)$*. Further,* $\mathtt{Setup}$ *draws an* $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_\ell$ *at random. It defines the sets as*

$$\begin{aligned} X &:= \mathbb{G}^{\ell+1}, \\ L &:= \langle [\mathbf{A}] \rangle = \{[\mathbf{x}] \in \mathbb{G}^{\ell+1} \mid \exists [\mathbf{w}] \in \mathbb{G}^\ell \colon [\mathbf{x}]_T = e([\mathbf{A}], [\mathbf{w}])\} \text{ and} \\ R &:= \{([\mathbf{x}], [\mathbf{w}]) \in \mathbb{G}^{\ell+1} \times \mathbb{G}^\ell \mid [\mathbf{x}]_T = e([\mathbf{A}], [\mathbf{w}])\}. \end{aligned}$$

$\mathtt{Setup}$ *finally returns* $(\mathbb{G}, \mathbb{G}_T, p, P, g_T, e)$ *and* $[\mathbf{A}]$ *as a compact description of* X, L *and* R*.*

Definition 4.4 (HPS for $\mathcal{D}_\ell$-MDDH with pairings)**.** *Let* $\mathtt{SMP}$ *the subset membership problem based on* $\mathcal{D}_\ell$*-MDDH with pairings as presented in Definition 4.3. It is straightforward to generalize the HPS from Definition 3.4. Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, p, P, g_T, e)$ *and a matrix* $[\mathbf{A}]$*,* $\mathtt{Setup}$ *defines* $\mathcal{HSK} := \mathbb{Z}_p^{\ell+1}$*,* $\Pi := \mathbb{G}_T$*,* $\mathcal{HPK} := \mathbb{G}$*. Further* $\mathtt{Setup}$ *defines*

$$\begin{aligned} \alpha \colon \mathbb{Z}_p^{\ell+1} &\to \mathbb{G}^{1\times\ell}, \mathsf{hsk} && \mapsto \mathsf{hsk}^\top \cdot [\mathbf{A}], \\ H_{\mathsf{hsk}} \colon \mathbb{G}^{\ell+1} &\to \mathbb{G}_T, \quad [\mathbf{x}] && \mapsto e([\mathsf{hsk}]^\top, [\mathbf{x}]) \\ F \colon \mathbb{G}^{\ell+1} \times \mathbb{G}_p^\ell \times \mathbb{G}^{1\times\ell} &\to \mathbb{G}_T, \quad ([\mathbf{x}], [\mathbf{w}], [\mathsf{hpk}]) && \mapsto e([\mathsf{hpk}], [\mathbf{w}]). \end{aligned}$$

Correctness, smoothness and efficiently checkable membership of the key space follow straightforward from the respective properties of the HPS defined in 3.4.

Note that in order to compute H_{hsk} *it is sufficient to know* $[\mathsf{hsk}] \in \mathbb{G}^{\ell+1}$*.*

Let $\ell \in \mathbb{N}$, $\ell \geq 2$ and $\mathtt{NIKE}$ our construction given in Figure 3, where $\mathtt{SMP}$ is instantiated with the SMP from Definition 4.3 and $\mathtt{HPS}$ with the corresponding hash proof system from Definition 4.4. Then we define $\mathtt{SMP}_{\mathsf{opt}}$ as follows. On input 1^λ, $\mathtt{SMP}_{\mathsf{opt}}.\mathtt{Setup}$ calls $\mathtt{SMP}.\mathtt{Setup}(1^\lambda)$ to obtain $[\mathbf{A}] \in \mathbb{G}^{(\ell+1)\times\ell}$. Further $\mathtt{SMP}_{\mathsf{opt}}.\mathtt{Setup}$ chooses $\mathbf{B} \overset{\$}{\leftarrow} \mathbb{Z}_p^{2(\ell+1)\times(\ell+1)}$ (if $\overline{\mathbf{B}}$ is not invertible it resamples) and defines

$$X_{\mathsf{opt}} := \mathbb{G}^{\ell+1} \times \mathbb{G}^\ell \times \mathbb{G}^{2\ell}$$

and

$$\begin{aligned} L_{\mathsf{opt}} := \{([\mathbf{x}], [\mathsf{hpk}], [\mathbf{c}]) \in X_{\mathsf{opt}} \mid \exists \mathbf{w}, \mathsf{hsk}, \mathbf{r} \colon [\mathbf{x}] = [\mathbf{A}] \cdot \mathbf{w} \wedge [\mathsf{hpk}] = \mathsf{hsk}^\top [\mathbf{A}] \\ \wedge [\mathbf{c}] = \begin{bmatrix} \mathbf{0} \\ \mathsf{hsk} \end{bmatrix} + [\mathbf{B}] \cdot \mathbf{r}\}. \end{aligned}$$

(We can view $C := [\mathbf{c}]$ as an encryption of $[\mathsf{hsk}]$ with randomness $r_{\mathsf{enc}} := \mathbf{r}$ and employ it for extraction) . Further, we can rewrite L_{opt} as

$$L_{\mathsf{opt}} := \{ \begin{bmatrix} \mathbf{x} \\ \mathsf{hpk} \\ \mathbf{c} \end{bmatrix} \in \mathbb{G}^{4\ell+1} \mid \exists \begin{pmatrix} \mathbf{w} \\ \mathsf{hsk} \\ \mathbf{r} \end{pmatrix} \in \mathbb{Z}_p^{3\ell+2} \colon \begin{bmatrix} \mathbf{x} \\ \mathsf{hpk} \\ \mathbf{c} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \overline{\mathbf{B}} \\ \mathbf{0} & \mathbf{1} & \underline{\mathbf{B}} \end{bmatrix}}_{[\mathbf{M}_{\mathsf{opt}}]:=} \cdot \begin{pmatrix} \mathbf{w} \\ \mathsf{hsk} \\ \mathbf{r} \end{pmatrix} \},$$

(R_{opt} accordingly) where $\mathbf{1} \in \mathbb{Z}_p^{(\ell+1)\times(\ell+1)}$ is the identity matrix and $\overline{\mathbf{B}}, \underline{\mathbf{B}} \in \mathbb{Z}_p^{(\ell+1)\times(\ell+1)}$ denote the upper and the lower part of $\mathbf{B}$ respectively. This allows us to employ a QANIZK for linear languages. We require the QANIZK to be one-time simulation sound, that is the adversary is only allowed to query the oracle $\mathcal{O}_{\mathsf{sim}}$ in the USS-experiment (see Figure 4) once. Roughly said, one-time simulation soundness suffices, as we only switch one public key to invalid. We recall an instantiation of such a proof system, namely the proof system of Kiltz and Wee [36], adapted to our setting, in Figure 11.

```
PS.Setup($1^\lambda, (X_{\mathsf{opt}}, L_{\mathsf{opt}}, R_{\mathsf{opt}})$):
  $\mathbf{V} \xleftarrow{\$} \mathcal{D}'_\ell$
  $\mathbf{K}_0, \mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{(4\ell+1)\times(\ell+1)}$
  $\mathsf{crs} := ([\mathbf{K}_0^\top \mathbf{M}_{\mathsf{opt}}], [\mathbf{K}^\top \mathbf{M}_{\mathsf{opt}}],$
          $[\mathbf{V}\mathbf{K}_0^\top], [\mathbf{V}\mathbf{K}^\top], [\mathbf{V}])$
  $\mathsf{trp} := (\mathbf{K}_0, \mathbf{K})$
  return $(\mathsf{crs}, \mathsf{trp})$

PS.Prove($\mathsf{crs}, \mathsf{ID}, [\mathbf{y}], \mathbf{w}$):
  parse $\mathsf{crs} =: ([\mathbf{K}_0^\top \mathbf{M}_{\mathsf{opt}}], [\mathbf{K}^\top \mathbf{M}_{\mathsf{opt}}],$
                  $[\mathbf{V}\mathbf{K}_0^\top], [\mathbf{V}\mathbf{K}^\top], [\mathbf{V}])$
  $\tau := \mathsf{H}(\mathsf{ID})$
  $[\Pi] := \left([\mathbf{K}_0^\top \mathbf{M}_{\mathsf{opt}}] + \tau[\mathbf{K}^\top \mathbf{M}_{\mathsf{opt}}]\right) \mathbf{w}$
  return $[\Pi]$

PS.Sim($\mathsf{crs}, \mathsf{trp}, \mathsf{ID}, [\mathbf{y}]$):
  parse $\mathsf{crs} =: ([\mathbf{K}_0^\top \mathbf{M}_{\mathsf{opt}}], [\mathbf{K}^\top \mathbf{M}_{\mathsf{opt}}],$
                  $[\mathbf{V}\mathbf{K}_0^\top], [\mathbf{V}\mathbf{K}^\top], [\mathbf{V}])$
  parse $\mathsf{trp} := (\mathbf{K}_0, \mathbf{K})$
  $\tau := \mathsf{H}(\mathsf{ID})$
  $[\Pi] := (\mathbf{K}_0 + \tau\mathbf{K})^\top [\mathbf{y}]$
  return $[\Pi]$

PS.Ver($\mathsf{crs}, \mathsf{ID}, [\mathbf{y}], [\Pi]$):
  parse $\mathsf{crs} =: ([\mathbf{K}_0^\top \mathbf{M}_{\mathsf{opt}}], [\mathbf{K}^\top \mathbf{M}_{\mathsf{opt}}],$
                  $[\mathbf{V}\mathbf{K}_0^\top], [\mathbf{V}\mathbf{K}^\top], [\mathbf{V}])$
  $\tau := \mathsf{H}(\mathsf{ID})$
  if $e([\mathbf{V}], [\Pi])$
      $= e([\mathbf{V}\mathbf{K}_0^\top] + \tau[\mathbf{V}\mathbf{K}^\top], [\mathbf{y}])$
    return 1
  else return 0
```

Figure 11: A one-time simulation-sound QANIZK PS for $\mathsf{SMP}_{\mathsf{opt}}$. Recall that L_{opt} is described via the matrix $\mathbf{M}_{\mathsf{opt}}$, where $\mathsf{H}\colon \mathcal{IDS} \to \mathbb{Z}_p$ is the output of a collision-resistant hash function generator $\mathcal{H}$. (Note that the public parameters have size $15\ell^2 + 11\ell + 1$ and the proofs have size $\ell + 1$.)

Theorem 4.5. *Let $\mathtt{NIKE}_{\mathsf{dkr}}$ be the NIKE from Figure 8, where PS is instantiated with the PS from Figure 11. Then $\mathtt{NIKE}_{\mathsf{dkr}}$ is DKR-CKS-heavy secure under the $\mathcal{D}'_\ell$-KMDH assumption in $\mathbb{G}$ and the $\mathcal{D}_\ell$-MDDH assumption in $\mathbb{G}$. More formally, for every adversary $\mathcal{A}$ with running time $t_\mathcal{A}$ we have adversaries $\mathcal{B}, \mathcal{B}_1, \mathcal{B}_2$ with running times $t_\mathcal{B} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2} \approx t_\mathcal{A}$ and*

$$\mathrm{Adv}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A},\mathit{NIKE}}(\lambda) \leq \mathrm{Adv}^{\mathsf{kmdh}}_{\mathcal{B},\mathcal{D}'_\ell\mathbb{G}}(\lambda) + \frac{1}{p} + \frac{q_{\mathsf{regH}}}{2}(\mathrm{Adv}^{\mathsf{cr}}_{\mathcal{B}_1,\mathcal{H}}(\mathrm{sec}) + \mathrm{Adv}^{\mathsf{smp}}_{\mathcal{B}_2,\mathit{SMP}}(\lambda) + \varepsilon),$$

where ε is negligible in λ and q_{regH} denotes the number of queries to $\mathcal{O}_{\mathsf{regH}}$ that $\mathcal{A}$ issues.

Sketch. By $\Pr[\mathbf{G}_i]$ we denote the probability that $\mathcal{A}$ wins game $\mathbf{G}_i$.

Game $\mathbf{G}_0$: The real experiment. Game $\mathbf{G}_0$ is the DKR-CKS-heavy experiment as presented in Figure 3, where $\mathcal{A}$ plays with a challenger $\mathcal{C}$. We have thus

$$\mathrm{Adv}^{\mathsf{dkr-cks-heavy}}_{\mathcal{A},\mathtt{NIKE}}(\lambda) = |\Pr[\mathbf{G}_0] - 1/2|\,.$$

Game $\mathbf{G}_1$: Guess the challenge. The modifications are exactly as in game $\mathbf{G}_1$ of the proof of Theorem 3.1. By perfect correctness of NIKE we have

$$\Pr[\mathbf{G}_1] = \Pr[\mathbf{G}_0].$$

Game $\mathbf{G}_2$: Recall that $\mathsf{sk}_{i^\star} = ([\mathbf{w}_{i^\star}], [\mathsf{hsk}_{i^\star}])$. Next, we want to make $[\mathbf{w}_{i^\star}]$ redundant also for dishonest reveal queries. In order to do so, for dishonest registration queries $(\mathsf{ID}, [\mathbf{x}, \mathsf{hpk}, \mathbf{c}], [\Pi])$ we will extract $[\mathsf{hsk}]$ from the proof such that $e([\mathbf{A}], [\mathsf{hsk}]) = e([\mathsf{hpk}], [1])$ and save $(\mathsf{ID}, [\mathbf{x}, \mathsf{hpk}, \mathbf{c}], [\Pi], [\mathsf{hsk}])$ in Q_{regC}. Whenever $[\mathbf{x}, \mathsf{hpk}, \mathbf{c}] \in L_{\mathsf{opt}}$ we can compute

$$[\mathsf{hsk}] = \underline{[\mathbf{c}]} - \underline{\mathbf{B}} \cdot \overline{\mathbf{B}}^{-1} \cdot \overline{[\mathbf{c}]},$$

where $\underline{[\mathbf{c}]}, \overline{[\mathbf{c}]} \in \mathbb{G}^{\ell+1}$ denote the upper and lower part of $[c]$.

Now, on a query $(\mathsf{ID}, \mathsf{ID}_{i^\star})$ to $\mathcal{O}_{\mathsf{revC}}$ with $(\mathsf{ID}, ([\mathbf{x}, \mathbf{c}], [\mathsf{hpk}]), [\Pi], [\mathsf{hsk}]) \in Q_{\mathsf{regC}}$ we compute the shared key as

$$H_{\mathsf{hsk}_{i^\star}}([\mathbf{x}]) \cdot H_{\mathsf{hsk}}([\mathbf{x}_{i^\star}]) \in \mathbb{G}_T.$$

Note that in order to compute $H_{\mathsf{hsk}}([\mathbf{x}_{i^\star}])$ it is sufficient to know $[\mathsf{hsk}]$.

By the correctness of the hash proof system, the answers of $\mathcal{O}_{\mathsf{revC}}$ in $\mathbf{G}_2$ are identical to the answers of $\mathcal{O}_{\mathsf{revC}}$ in $\mathbf{G}_1$, whenever $[\mathbf{x}, \mathsf{hpk}, \mathbf{c}] \in L_{\mathsf{opt}}$.

By bad we denote the event that extraction is not successful for at least one query (which is only the case if the adversary managed to forge a proof for a statement outside the language). In this case we return 1 with probability 1/2 and abort. Soundness of $\mathtt{PS}$ under $\mathcal{D}'_\ell$-KMDH yields

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \leq \mathrm{Adv}^{\mathsf{kmdh}}_{\mathcal{B},\mathcal{D}'_\ell\mathbb{G}}(\lambda) + 1/p.$$

(For more details on the bound we refer to [36].)

Game $\mathbf{G}_3$: Abort upon wrong guess. We change the winning condition of the game as follows. If $\mathsf{ID}_{i^\star}$ is not included in the test query of $\mathcal{A}$, the experiment returns 1 with probability 1/2 and aborts. Then it holds

$$\begin{aligned}\Pr[\mathbf{G}_3] &= \Pr[\mathbf{G}_2] \cdot 2/q_{\mathsf{regH}} + 1/2 \cdot (1 - 2/q_{\mathsf{regH}}) \\ &= (\Pr[\mathbf{G}_2] - 1/2) \cdot 2/q_{\mathsf{regH}} + 1/2\end{aligned}$$

and thus

$$\Pr[\mathbf{G}_2] - 1/2 = q_{\mathsf{regH}}/2 \cdot (\Pr[\mathbf{G}_3] - 1/2).$$

Game $\mathbf{G}_4$: From Game $\mathbf{G}_4$ on we abort if after registering $\mathsf{ID}_{i^\star}$ honestly there is a dishonest key registration query $(\mathsf{ID}, \cdot, \cdot)$ with $\mathsf{H}(\mathsf{ID}) = \mathsf{H}(\mathsf{ID}_{i^\star})$. The collision resistance of $\mathcal{H}$ yields

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_4]| \leq \mathrm{Adv}^{\mathsf{cr}}_{\mathcal{B}_1,\mathcal{H}}(\lambda).$$

Game $\mathbf{G}_5$: Remove the secret key. Upon receiving the $i^\star$-th register honest user query, $\mathcal{C}$ deviates from the $\mathtt{NIKE.KeyGen}$ procedure as follows: The challenger $\mathcal{C}$ draws $[\mathbf{x}_{i^\star}] \overset{\$}{\leftarrow} X_{\mathtt{SMP}} \setminus L_{\mathtt{SMP}}$. $\mathcal{C}$

A distinguisher between both games can be turned directly into a SMP attacker $\mathcal{B}$ putting his challenge in the place of $[\mathbf{x}_{i^\star}]$. If the challenge was in L, Game $\mathbf{G}_4$ was simulated, else it was Game $\mathbf{G}_5$. Observe that it is crucial here that $\mathcal{C}$ does not make use of $[\mathbf{w}_{i^\star}]$ anymore due to the changes made in the previous games. Because of the changes in game $\mathbf{G}_4$, we ensure that the proof $[\Pi_{i^\star}]$ cannot be re-used for a dishonest registration query.

Altogether this yields

$$|\Pr[\mathbf{G}_4] - \Pr[\mathbf{G}_5]| \leq \mathrm{Adv}^{\mathsf{smp}}_{\mathcal{B}_2,\mathtt{SMP}}(\lambda).$$

Game $\mathbf{G}_6$: Randomize the test query. As in CKS-heavy we can now randomize the test-query and obtain

$$|\Pr[\mathbf{G}_5] - \Pr[\mathbf{G}_6]| \leq \varepsilon.$$

Finally, $\mathbf{G}_6$ does not depend on the challenge bit anymore, and thus $\Pr[\mathbf{G}_6] = 1/2$. Collecting the advantages proves the theorem.

□

5 Optimality of our construction

Our NIKE scheme in Section 3 does not meet the lower bound regarding tightness proven in [4]. We can circumvent their result since our scheme does not offer a public and efficient algorithm for checking validity of public keys (called `PKCheck` in [4]): the reduction introduces invalid public keys where the statement is not from the language. It follows from the hardness of the subset membership problem that this is not detectable given *just the public key.*

This immediately raises the question whether, in this new setting without efficient and public `PKCheck`, we can still obtain a lower bound for the tightness of *HKR - CKS - heavy*-secure NIKE schemes. We answer

```
$\mathrm{Exp}^{\mathsf{uf-cks-heavy}}_{\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2),n,\mathtt{NIKE}}(\lambda)$:
  $\mathcal{PP} \xleftarrow{\$} \mathtt{NIKE.Setup}(1^\lambda)$
  $\mathsf{ID}_1, ..., \mathsf{ID}_n \xleftarrow{\$} \mathcal{IDS}$ (all disjoint)
  $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}_i), i = 1, ..., n$
  $(st, \{i^\star, j^\star\}) \leftarrow \mathcal{A}_1(\mathcal{PP}, \mathsf{ID}_1, \mathsf{pk}_1, ..., \mathsf{ID}_n, \mathsf{pk}_n)$
  $K^\star \leftarrow \mathcal{A}_2(st, (\mathsf{sk}_i)_{i\in[n]\setminus\{i^\star,j^\star\}})$
  if $K^\star = \mathtt{NIKE.SharedKey}(\mathsf{ID}_{i^\star}, \mathsf{pk}_{i^\star}, \mathsf{ID}_{j^\star}, \mathsf{sk}_{j^\star})$
    then output 1
  else output 0
```

Figure 12: Experiment for $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ security of a NIKE scheme NIKE with shared key space $\mathcal{K}$, for any $n \in \mathbb{N}$. The set $C := \{i^\star, j^\star\}$ contains the indices of the two public keys $\mathcal{A}$ wishes to be challenged upon. The set $[n] \setminus C$ contains all indices of the $n-2$ public keys for which $\mathcal{A}$ learns a secret key from the experiment.

this question in the affirmative and prove a new lower bound that meets the loss of our reduction in Section 3. To present our result, we first give some definitions.

Since *HKR - CKS - heavy* security provides several oracles to the adversary which can be queried in an arbitrary order, a reduction to *HKR - CKS - heavy*-security cannot be formalized as an algorithm in a short and easy way. As done in previous impossibility results before, we thus prove our result w.r.t a weaker security notion that is easier to present. Afterwards, we show that our result carries over to *HKR - CKS - heavy*-security. Our weaker notion is called $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$[6]. The security experiment is depicted in Figure 12. Observe that the experiment provides the adversary with all but two secret keys, and thus implicitly with all but one shared key. The adversary chooses which keys he wants to see after obtaining all public keys in the system. The notion is further weakened by letting the number of users in the system be a fixed $n \in \mathbb{N}$ instead of letting the adversary determine it on-the-fly (i.e., via $\mathcal{O}_{\mathsf{regH}}$ queries).

The next lemma allows us to prove a lower bound w.r.t $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ instead of *HKR - CKS - heavy*. It will become crucial that the reduction is tight.

Lemma 5.1 (*HKR - CKS - heavy* $\Rightarrow$ $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$)**.** *For every adversary $\mathcal{A}$ attacking $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ in running time $t_\mathcal{A}$ with success probability $\varepsilon_\mathcal{A}$, there exists an adversary $\mathcal{B}$ attacking CKS - heavy in running time $t_\mathcal{B} \approx t_\mathcal{A}$ and success probability $\varepsilon_\mathcal{B} = \varepsilon_\mathcal{A}$.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ adversary. We show how to construct a *HKR - CKS - heavy* adversary $\mathcal{B}$.

On input $\mathcal{PP}$ by the challenger, the adversary $\mathcal{B}$ first generates random, disjoint identities $\mathsf{ID}_1, ..., \mathsf{ID}_n$ and calls the oracle $\mathcal{O}_{\mathsf{regH}}(\mathsf{ID}_i)$ for all $i \in [n]$. $\mathcal{B}$ thus obtains $\mathsf{pk}_1 ..., \mathsf{pk}_n$. Now, $\mathcal{B}$ runs $\mathcal{A}_1(\mathcal{PP}, \mathsf{pk}_1, ..., \mathsf{pk}_n)$ and obtains a state $st_\mathcal{A}$ and a set $C := \{i^\star, j^\star\}$. Now, for every $i \in [n] \setminus C$, $\mathcal{B}_1$ queries its oracle $\mathcal{O}_{\mathsf{extr}}(\mathsf{ID}_i)$ which returns a secret key sk_i. Next, $\mathcal{B}_1$ runs $\mathcal{A}_2(st_\mathcal{A}, (\mathsf{sk}_i)_{i\in[n]\setminus C})$ and obtains a key $K^\star$. The adversary $\mathcal{B}$ finally queries its test oracle on $(\mathsf{ID}_{i^\star}, \mathsf{ID}_{j^\star})$ which returns a key K. It outputs 0 if $K^\star = K$ and 1 otherwise. As we assume the shared key to be uniquely determined and as further $\mathcal{B}$ only queries $\mathcal{O}_{\mathsf{extr}}$ on identities ID_i with $i \notin C$ we obtain $\varepsilon_\mathcal{B} = \varepsilon_\mathcal{A}$. $\square$

We recall the definition of a non-interactive complexity assumption, taken verbatim from [4], Def. 4 and 5.

Definition 5.2 (Non-interactive complexity assumption)**.** *A non-interactive complexity assumption (NICA) $N = (T, V, U)$ consists of three turing machines. The instance generation machine $(c, w) \xleftarrow{\$} T(1^\lambda)$ 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.*

[6]We work with an even weaker notion that [4]. The main difference is that our adversary only has a secret key oracle (from which it can compute shared keys itself), while the adversary in [4] is provided with a shared key oracle.

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

Figure 13: Security experiment for a non-interactive complexity assumption (NICA).

Definition 5.3. *We say that $\mathcal{B}$ (t,ε)-breaks a NICA $N=(T,U,V)$ if $\mathcal{B}$ runs in time $t(\lambda)$ and it holds that*

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

where $\mathrm{Exp}^{\mathsf{nica}}_{\mathcal{B},N}$ is the experiment defined in Figure 13 and the probability is taken over the random coins consumed by T and the uniformly random choices in the experiment.

Now we are ready to formalize what we mean by a reduction Λ from a NICA to the $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ security of `NIKE`. We closely follow the structure of [4] and similar to [12, 30, 33, 37, 4] only consider a certain class of reductions.

Definition 5.4 (Simple reduction)**.** *We call a TM Λ a $(t_\Lambda, n, \varepsilon_\Lambda, \varepsilon_\mathcal{A})$-*reduction *from breaking a NICA $N=(T,U,V)$ to breaking the $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ security of `NIKE`, if Λ turns an adversary $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ that runs in time $t_\mathcal{A}$ and has advantage $\varepsilon_\mathcal{A}$ to break $\mathrm{Exp}^{\mathsf{uf-cks-heavy}}_{\mathcal{A},n,\mathtt{NIKE}}$ (as provided in Figure 12) into a TM $\mathcal{B}$ that runs in time $t_\Lambda + t_\mathcal{A}$ and has advantage ε_Λ to break N (see Definition 5.3). We call Λ* simple*, if Λ has only black-box access to $\mathcal{A}$ and executes $\mathcal{A}$ only once (and in particular without rewinding).*

In the following we will only consider *simple* reductions. Note that even though this seems to restrict the class of reductions heavily, actually most reductions (including reductions performing hybrid steps) are simple. The security proofs of all existing NIKE schemes [15, 10, 18] we are aware of[7] are simple reductions.

Since our notion of $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$-security requires only two rounds of interaction between the adversary and the challenger, we are able to give a very compact formal description of the algorithm $\Lambda := (\Lambda_1, \Lambda_2, \Lambda_3)$ as follows:

- Λ_1 is a probabilistic algorithm that gets as input a (set of) NICA challenge(s) c and outputs public parameters $\mathcal{PP}$, a set of identities and public keys $\mathsf{ID}_1, \mathsf{pk}_1, ..., \mathsf{ID}_n, \mathsf{pk}_n$ and a state st_1.
- Λ_2 is a deterministic algorithmn that receives as input $C \subseteq [n]$ with $|C| = 2$ (else aborts) and st_1 and outputs $(st_2, (sk_i)_{i\in[n]\setminus C})$.
- Λ_3 is a deterministic algorithm that receives as input st_2 and $\tilde{K}$ and outputs an s.

5.1 A weaker validity check

We expand the results from [4] by relaxing the assumptions on the publicly checkable validity of public keys. Recall that [4] requires a method `PKCheck` allowing to efficiently verify whether a public key pk was generated by $\mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID})$, e.g., whether there exists a secret key sk and random coins r such that $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}; r)$. We will only require the following notion of weak checkability of public keys. In particular, we only require it to be checkable whether a public key is valid *given a corresponding secret key.*

Definition 5.5. *Let `NIKE` be a NIKE with secret key space $\mathcal{SK}$, identity space $\mathcal{IDS}$ and public key space $\mathcal{PK}$. We say that `NIKE` satisfies* weak checkability of public keys*, if there exists a efficiently computable function*

$$\mathtt{wPKCheck}\colon \mathcal{IDS} \times \mathcal{PK} \times \mathcal{SK} \rightarrow \{0,1\}$$

[7]Remember that we restrict to 2-party key exchange protocols in the setting where a PKI is available.

with the following properties:

$$\textit{For all } (\mathsf{pk},\mathsf{sk}) \leftarrow \texttt{NIKE.KeyGen}(\mathcal{PP},\mathsf{ID}) \textit{ we have } \texttt{wPKCheck}(\mathsf{ID},\mathsf{pk},\mathsf{sk}) = 1. \tag{1}$$

$$\begin{aligned} &\textit{For all } (\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{sk}_1),(\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{sk}_1'),(\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{sk}_2) \textit{ with } \texttt{wPKCheck}(\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{sk}_1)\\ &= \texttt{wPKCheck}(\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{sk}_1') = \texttt{wPKCheck}(\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{sk}_2) = 1 \textit{ it holds}\\ &\texttt{NIKE.SharedKey}(\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{ID}_1,\mathsf{sk}_1) = \texttt{NIKE.SharedKey}(\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{ID}_1,\mathsf{sk}_1'). \end{aligned} \tag{2}$$

We call a secret key sk valid *for* $(\mathsf{ID},\mathsf{pk})$ *if* $\texttt{wPKCheck}(\mathsf{ID},\mathsf{pk},\mathsf{sk}) = 1$. *We further define the* language of valid public keys

$$L^{\text{valid}} := \{(\mathsf{ID},\mathsf{pk}) \mid \exists \mathsf{sk} \colon \texttt{wPKCheck}(\mathsf{ID},\mathsf{pk},\mathsf{sk}) = 1\}.$$

Property 2 now implies that any two tuples $(\mathsf{ID}_1,\mathsf{pk}_1),(\mathsf{ID}_2,\mathsf{pk}_2) \in L^{\text{valid}}$ *lead to a unique shared key independently of which valid secret key is employed to compute the shared key.*

Remark 5.6. *Note that a NIKE for which it can be efficiently verified whether a pair* $(\mathsf{pk},\mathsf{sk})$ *lies in the image of* $\texttt{NIKE.KeyGen}(\mathcal{PP},\mathsf{ID})$ *in particular satisfies weak checkability of public keys with*

$$\texttt{wPKCheck}(\mathsf{ID},\mathsf{pk},\mathsf{sk}) = \begin{cases} 1 & \textit{if } \exists r : (\mathsf{pk},\mathsf{sk}) = \texttt{NIKE.KeyGen}(\mathcal{PP},\mathsf{ID};r)\\ 0 & \textit{else} \end{cases}.$$

5.2 A lower bound on tightness

In this section we show that if a NIKE $\texttt{NIKE}$ satisfies weak checkable uniqueness, then any simple reduction from a NICA to the $\mathit{UF\text{-}CKS\text{-}heavy}_n$-security of $\texttt{NIKE}$ it has to inherently lose a factor of $n/2$ in reduction, where n is the number of public keys. Further, we show that the NIKE $\texttt{NIKE}$ presented in Figure 6 satisfies weak checkability of public keys. Note that by definition any NIKE supporting weak checkability of public keys is *perfectly correct*, that is for all $(\mathsf{ID}_i,\mathsf{pk}_i,\mathsf{sk}_i) \xleftarrow{\$} \texttt{NIKE.KeyGen}(\mathcal{PP},\mathsf{ID}_i), i \in \{1,2\}$, we have

$$\texttt{NIKE.SharedKey}(\mathsf{ID}_1,\mathsf{pk}_1,\mathsf{ID}_2,\mathsf{sk}_2) = \texttt{NIKE.SharedKey}(\mathsf{ID}_2,\mathsf{pk}_2,\mathsf{ID}_1,\mathsf{sk}_1).$$

Theorem 5.7. *Let* $N = (T,U,V)$ *be a non-interactive complexity assumption and* $n \in \mathrm{poly}(\lambda)$. *Let* $\texttt{NIKE}$ *be a* $\mathit{UF\text{-}CKS\text{-}heavy}_n$ *secure NIKE with shared key space* $\mathcal{K}$, *public key space* $\mathcal{PK}$ *and secret key space* $\mathcal{SK}$ *which satisfies weak checkability of public keys via algorithm* $\texttt{wPKCheck}$. *Let further evaluating* $\texttt{wPKCheck}$ *require time* t_{wPKCheck}. *Then any reduction* $\Lambda = (\Lambda_1,\Lambda_2,\Lambda_3)$ *from* N *to* $\texttt{NIKE}$ *has to lose a factor* $n/2$ *assuming* N *is hard. More formally, for any simple* $(t_\Lambda, n, \varepsilon_\Lambda, 1)$*-reduction from breaking the assumption* N *to breaking the* $\mathit{UF\text{-}CKS\text{-}heavy}_n$*-security of* $\texttt{NIKE}$, *there exists an adversary* $\mathcal{B}$ *breaking* N *in running time*

$$t_{\mathcal{B}} \le \frac{n(n-1)}{2} t_\Lambda + \frac{n(n-1)(n-2)}{2} t_{\text{wPKCheck}}$$

with success probability

$$\varepsilon_{\mathcal{B}} \ge \varepsilon_\Lambda - \frac{2}{n}.$$

Remark 5.8. *We have* $\varepsilon_{\mathcal{A}} = 1$ *and* $\varepsilon_{\mathcal{B}} = \eta(\lambda)$ *for a negligible function* η *(as* N *is assumed to be hard). We can thus transform the last equation into* $\varepsilon_\Lambda \le \frac{2}{n}\varepsilon_{\mathcal{A}} + \eta(\lambda)$. *This implies the claimed reduction loss of* $n/2$.

Proof. We follow the proof structure of [30], [37], [4].

THE HYPOTHETICAL ADVERSARY. In the following we describe a hypothetical adversary $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$. Note that this adversary might not be efficient, but in order to prove the reduction loss of $n/2$ we show how to simulate it efficiently.

$\mathcal{A}_1(\mathcal{PP},\mathsf{ID}_1,\mathsf{pk}_1,\ldots,\mathsf{ID}_n,\mathsf{pk}_n)$ chooses $C := \{i^\star, j^\star\} \subseteq [n]$ with $|C| = 2$ uniformly at random. It outputs (st, C), where $st = (\mathcal{PP},\mathsf{ID}_1,\mathsf{pk}_1,\ldots,\mathsf{ID}_n,\mathsf{pk}_n,C)$.

$\mathcal{A}_2(st,(\mathsf{sk}_i)_{i\in[n]\setminus C})$ checks whether $\texttt{wPKCheck}(\mathsf{ID}_i,\mathsf{pk}_i,\mathsf{sk}_i) = 1$ for all $i \in [n] \setminus C$ and whether $(\mathsf{ID}_i,\mathsf{pk}_i) \in L^{\text{valid}}$ for both $i \in C$. If this is the case $\mathcal{A}_2$ computes a secret key $\mathsf{sk}_{j^\star}$ s.t. $\texttt{wPKCheck}(\mathsf{ID}_{j^\star},\mathsf{pk}_{j^\star},\mathsf{sk}_{j^\star}) = 1$ and outputs $K^\star = \texttt{NIKE.SharedKey}(\mathsf{ID}_{i^\star},\mathsf{pk}_{i^\star},\mathsf{ID}_{j^\star},\mathsf{sk}_{j^\star})$. Otherwise $\mathcal{A}_2$ outputs $\perp$.

As we have $(\mathsf{ID}, \mathsf{pk}, \mathsf{sk}) \in \mathcal{R}^{\text{unique}}$ for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \texttt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID})$ and further $\texttt{NIKE.SharedKey}$ returns a unique key for all tuples passing $\texttt{wPKCheck}$, due to property 2 of Definition 5.5 the hypothetical adversary always wins in the $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ experiment.

We now describe an adversary $\mathcal{B}$ attempting to break $N = (T, U, V)$. The strategy is to run the reduction $\Lambda = (\Lambda_1, \Lambda_2, \Lambda_3)$ simulating $\mathcal{A}$ efficiently. Let c be the input of $\mathcal{B}$, where $(c, w) \leftarrow T(1^\lambda)$. Let $SK[\,], SK^\star[\,]$ be arrays of n entries initialized by $\emptyset$ and maintained throughout the reduction by $\mathcal{B}$.

1. The adversary $\mathcal{B}$ runs $(st_1, \mathcal{PP}, \mathsf{ID}_1, \mathsf{pk}_1, \ldots, \mathsf{ID}_n, \mathsf{pk}_n) \leftarrow \Lambda_1(c)$.
2. The adversary $\mathcal{B}$ samples $\{i^\star, j^\star\} = C^\star \subset [n]$ with $|C^\star| = 2$ uniformly at random.
3. For each $C \subset [n]$ with $|C| = 2$ the adversary $\mathcal{B}$ runs the reduction $\Lambda_2(st_1, C)$. Let $(st_2^C, (\mathsf{sk}_i^C)_{i \in [n] \setminus C})$ denote the output of the respective execution. Whenever $\texttt{wPKCheck}(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{sk}_i^C) = 1$ for an $i \in [n] \setminus C$ the adversary sets $SK[i] = \mathsf{sk}_i^C$. If $C = C^\star$, $\mathcal{B}$ additionally sets $SK^\star[i] = \mathsf{sk}_i^{C^\star}$
4. If there exists an $i \in [n] \setminus C^\star$ with $SK^\star[i] = \emptyset$ (i.e. $\texttt{wPKCheck}(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{sk}_i^{C^\star}) = 0$) or there exists a $i \in C^\star$ such that $SK[i] = \emptyset$ (i.e. $\texttt{wPKCheck}(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{sk}_i^C) = 0$ for all $C \subseteq [n]$ with $|C| = 2$) then $\mathcal{B}$ sets $K^\star = \perp$. Otherwise $\mathcal{B}$ computes $K^\star = \texttt{NIKE.SharedKey}(\mathsf{ID}_{i^\star}, \mathsf{pk}_{i^\star}, \mathsf{ID}_{j^\star}, SK[j^\star])$.
5. Finally, the adversary $\mathcal{B}$ outputs $s \xleftarrow{\$} \Lambda_3(st_2^{C^\star}, C^\star, K^\star)$.

Efficiency of $\mathcal{B}$. In the third step Λ_2 has to be executed $\binom{n}{2} = \frac{n(n-1)}{2}$ times. Each time the validity check has to be performed $n-2$ times. For the running time of $\mathcal{B}$ it thus holds

$$t_{\mathcal{B}} \leq \frac{n(n-1)}{2} t_\Lambda + \frac{n(n-1)(n-2)}{2} t_{\text{wPKCheck}}.$$

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

$$\begin{aligned} \mathsf{check\text{-}fails}: &\quad \exists i \in [n] \setminus C^\star \text{ such that } SK^\star[i] = \emptyset \\ \mathsf{pk\text{-}valid}: &\quad \forall i \in \mathcal{C}^\star \text{ it holds that } SK[i] \neq \emptyset \end{aligned}$$

We first want to show that in the case of $\mathsf{check\text{-}fails} \vee \mathsf{pk\text{-}valid}$, $\mathcal{B}$ simulates the hypothetical adversary $\mathcal{A}$ perfectly. If $\mathsf{check\text{-}fails}$ occurs, then $\mathcal{B}$ returns $\perp$. The hypothetical adversary would have returned $\perp$ as well because in this case it holds $\texttt{wPKCheck}(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{sk}_i^{C^\star}) = 0$ for an $i \in [n] \setminus C^\star$. If $\mathsf{pk\text{-}valid}$ occurs, we have $(\mathsf{ID}_i, \mathsf{pk}_i) \in L^{\text{valid}}$ for all $i \in [n]$ (as in this case for each $i \in [n]$ there exists a set $C \subset [n]$ such that the reduction Λ_2 provided an sk_i^C with $\texttt{wPKCheck}(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{sk}_i^C) = 1$ at some point). In this case the shared key $K^\star$ is unique by property 1 in Definition 5.5 and can be computed by $\mathcal{B}$ with the secret key $\mathcal{SK}[j^\star]$.

We summarize all other possible cases in the event

$$\mathsf{bad} = \neg\mathsf{check\text{-}fails} \wedge \neg\mathsf{pk\text{-}valid},$$

which is well-defined, as Λ_2 is deterministic.

We now bound the probability that bad happens. For this, we observe that $\neg\mathsf{pk\text{-}valid}$ can only occur if the event $E := (\exists i \in [n] \text{ s.t. } \mathcal{SK}[i] = \emptyset)$ occurs. As $C^\star$ is chosen uniformly at random and the view of Λ_2 is independent of $C^\star$, we have $i \in [n] \setminus C^\star$ with probability $1 - 2/n$. In this case $\mathsf{check\text{-}fails}$ occurs and thus $\Pr[\mathsf{check\text{-}fails} |\, E] \geq 1 - 2/n$. Now since $\neg\mathsf{pk\text{-}valid} \Rightarrow E$ it holds that $\Pr[\neg\mathsf{check\text{-}fails} \wedge \neg\mathsf{pk\text{-}valid}] \leq \Pr[\neg\mathsf{check\text{-}fails} \wedge E] = \Pr[\neg\mathsf{check\text{-}fails}|E] \cdot \Pr[E] \leq \Pr[\neg\mathsf{check\text{-}fails}|E] = 1 - \Pr[\mathsf{check\text{-}fails}|E] \leq 2/n$. We thus obtain

$$\Pr[\mathsf{bad}] \leq 2/n.$$

Let $\varepsilon_{\mathcal{B}}\big|_{\neg\mathsf{bad}}$ denote the probability of $\mathcal{B}$ to win under the condition that bad does not occur and $\varepsilon_\Lambda\big|_{\neg\mathsf{bad}}$ accordingly. We have

$$|\varepsilon_{\mathcal{B}} - \varepsilon_\Lambda| \leq \left|\varepsilon_{\mathcal{B}}\big|_{\neg\mathsf{bad}} - \varepsilon_\Lambda\big|_{\neg\mathsf{bad}}\right| + \Pr[\mathsf{bad}] = \Pr[\mathsf{bad}] \leq \frac{2}{n}.$$

□

Remark 5.9. *As shown in [4] it is straightforward to generalize Theorem 5.7 to simple $(t_\Lambda, n, \varepsilon_\Lambda, \varepsilon_\mathcal{A})$-reductions for general $\varepsilon_\mathcal{A}$ by letting the hypothetical adversary (and $\mathcal{B}$ respectively) toss a coin and only return $K^\star$ with probability $\varepsilon_\mathcal{A}$.*

Remark 5.10. *While Theorem 5.7 establishes the impossibility of tight security reductions for a large class of NIKE schemes, it thereby also gives a hint about how a tight NIKE scheme has to be constructed. Namely, such a scheme has to violate the assumptions made in the theorem such as the existence of an efficient* `PKCheck` *that, given the secret key, decides uniqueness of shared keys. More detailed, a tight NIKE scheme needs to allow a reduction to indistinguishably switch public keys to invalid (in fact, even* tightly *switch many of them in one step), such that invalid public keys admit* many *secret keys that lead do* different *shared keys. It is an interesting open question how to construct such a scheme.*

5.3 Weak checkable uniqueness of our NIKE

Lemma 5.11. *If instantiated with a hash proof system* `HPS` *where membership in $\mathcal{HSK}$ is efficient checkable for all sets of secret keys in the image of* `HPS.Setup`*, the NIKE* `NIKE` *presented in Figure 6 complies with weak checkability of public keys.*

Proof. Let $\mathcal{PP} := ((X, L, R), \mathcal{HSK}, \mathcal{H}, \alpha, F) \xleftarrow{\$} \mathtt{NIKE.Setup}(1^\lambda)$. We define

$$\mathtt{wPKCheck}(\mathsf{ID}, (\mathsf{hpk}, x), (\mathsf{hsk}, x, w)) := \begin{cases} 1 & \text{if } \mathsf{hsk} \in \mathcal{HSK} \wedge \alpha(\mathsf{hsk}) = \mathsf{hpk} \\ & \quad \wedge (x, w) \in R \\ 0 & \text{else} \end{cases}.$$

We have to show that `wPKCheck` is efficiently computable and further that `wPKCheck` meets properties 1 and 2 in Definition 5.5. By prerequisites we have that membership in $\mathcal{HSK}$ is efficiently checkable. Further, by definition of a hash proof system the map α and the relation R are efficiently computable. Property 1 follows straightforward from the definition of `wPKCheck`. Note that actually we have equality, that is

$$\mathtt{wPKCheck}(\mathsf{ID}, \mathsf{pk}, \mathsf{sk}) = 1 \Leftrightarrow \exists r \colon (\mathsf{pk}, \mathsf{sk}) = \mathtt{NIKE.KeyGen}(\mathcal{PP}, \mathsf{ID}; r).$$

It remains to prove prop. 2: for all $(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{sk}'_1), (\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{sk}_2)$ that all pass `wPKCheck` we have

$$\mathtt{NIKE.SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1) = \mathtt{NIKE.SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}'_1).$$

Let in the following $\mathsf{pk}_1 =: (\mathsf{hpk}_1, x_1), \mathsf{pk}_2 =: (\mathsf{hpk}_2, x_2), \mathsf{sk}_1 =: (\mathsf{hsk}_1, w_1), \mathsf{sk}'_1 =: (\mathsf{hsk}'_1, w'_1)$ and $\mathsf{sk}_2 =: (\mathsf{hsk}_2, w_2)$. By the properties of the hash proof system we have that for $\mathsf{hsk}_1, \mathsf{hsk}'_1 \in \mathcal{HSK}$ with $\alpha(\mathsf{hsk}_1) = \alpha(\mathsf{hsk}'_1) = \mathsf{hpk}_1$ and $x_2 \in L$ it holds

$$H_{\mathsf{hsk}_1}(x_2) = F(x_2, w_2, \mathsf{hpk}) = H_{\mathsf{hsk}'_1}(x_2)$$

and for w'_1 with $(x_1, w'_1) \in \mathcal{R}$ it holds

$$F(x_1, w_1, \mathsf{hpk}_2) = H_{\mathsf{hsk}_2}(x_1) = F(x_1, w'_1, \mathsf{hpk}_2).$$

This yields

$$\begin{aligned} \mathtt{NIKE.SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1) &= H_{\mathsf{hsk}_1}(x_2) \oplus F(x_1, w_1, \mathsf{hpk}_2) \\ &= H_{\mathsf{hsk}'_1}(x_2) \oplus F(x'_1, w'_1, \mathsf{hpk}_2) \\ &= \mathtt{NIKE.SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}'_1). \end{aligned}$$

□

Corollary 5.12 (Informal). *The security reduction in the proof of Theorem 3.1 is optimal regarding tightness among all simple reductions.*

Proof. Theorem 5.7 shows that simple security reductions for a NIKE admitting a weak PKCheck encounter a loss of at least $n/2$. Lemma 5.11 proves that our NIKE admits such a weak PKCheck and thus from Theorem 5.7 it follows that $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$-security of our NIKE can only be shown by a simple reduction if the reduction loses at least a factor of $n/2$. Now Lemma 5.1 shows that a $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ adversary tightly implies a $\mathit{HKR}\text{-}\mathit{CKS}\text{-}\mathit{heavy}$ adversary. Thus, any reduction with loss M from a NICA to $\mathit{HKR}\text{-}\mathit{CKS}\text{-}\mathit{heavy}$ security would imply a reduction with loss M to $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ security. It follows that $M \geq n/2$. □

Remark 5.13. *Since DKR-CKS-heavy security also tightly implies $\mathit{UF}\text{-}\mathit{CKS}\text{-}\mathit{heavy}_n$ security, our result carries over to DKR-CKS-heavy secure NIKE schemes that comply with weak checkable uniqueness.*

Acknowledgements

We would like to thank Kenny Paterson for collaboration at early stages of this work. We would like to thank the anonymous reviewers of Crypto 2018 for their helpful comments. Also, we would like to thank Eike Kiltz for pointing out a mistake in a previous version of Figure 1.

References

[1] Masayuki Abe et al. "Compact Structure-Preserving Signatures with Almost Tight Security". In: *Advances in Cryptology – CRYPTO 2017, Part II.* Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2017, pp. 548–580.

[2] Masayuki Abe et al. "Tagged One-Time Signatures: Tight Security and Optimal Tag Size". In: *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography.* Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. Lecture Notes in Computer Science. Nara, Japan: Springer, Heidelberg, Germany, 2013, pp. 312–331. DOI: 10.1007/978-3-642-36362-7_20.

[3] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. "A Framework for Identity-Based Encryption with Almost Tight Security". In: *Advances in Cryptology – ASIACRYPT 2015, Part I.* Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. Lecture Notes in Computer Science. Auckland, New Zealand: Springer, Heidelberg, Germany, 2015, pp. 521–549. DOI: 10.1007/978-3-662-48797-6_22.

[4] Christoph Bader et al. "On the Impossibility of Tight Cryptographic Reductions". In: *Advances in Cryptology – EUROCRYPT 2016, Part II.* Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, 2016, pp. 273–304. DOI: 10.1007/978-3-662-49896-5_10.

[5] Christoph Bader et al. "Tightly-Secure Authenticated Key Exchange". In: *TCC 2015: 12th Theory of Cryptography Conference, Part I.* Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, 2015, pp. 629–658. DOI: 10.1007/978-3-662-46494-6_26.

[6] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. "Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements". In: *Advances in Cryptology – EUROCRYPT 2000.* Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Bruges, Belgium: Springer, Heidelberg, Germany, 2000, pp. 259–274.

[7] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. "(Hierarchical) Identity-Based Encryption from Affine Message Authentication". In: *Advances in Cryptology – CRYPTO 2014, Part I.* Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2014, pp. 408–425. DOI: 10.1007/978-3-662-44371-2_23.

[8] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th Annual ACM Symposium on Theory of Computing.* Chicago, IL, USA: ACM Press, 1988, pp. 103–112.

[9] Dan Boneh and Ramarathnam Venkatesan. "Breaking RSA May Not Be Equivalent to Factoring". In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer, Heidelberg, Germany, 1998, pp. 59–71.

[10] David Cash, Eike Kiltz, and Victor Shoup. "The Twin Diffie-Hellman Problem and Applications". In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Istanbul, Turkey: Springer, Heidelberg, Germany, 2008, pp. 127–145.

[11] Jie Chen and Hoeteck Wee. "Fully, (Almost) Tightly Secure IBE and Dual System Groups". In: *Advances in Cryptology – CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2013, pp. 435–460. DOI: `10.1007/978-3-642-40084-1_25`.

[12] Jean-Sébastien Coron. "Optimal Security Proofs for PSS and Other Signature Schemes". In: *Advances in Cryptology – EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, 2002, pp. 272–287.

[13] Ronald Cramer and Victor Shoup. "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption". In: *Advances in Cryptology – EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, 2002, pp. 45–64.

[14] Alfredo De Santis et al. "Robust Non-interactive Zero Knowledge". In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2001, pp. 566–598.

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

[16] Alex Escala et al. "An Algebraic Framework for Diffie-Hellman Assumptions". In: *Advances in Cryptology – CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2013, pp. 129–147. DOI: `10.1007/978-3-642-40084-1_8`.

[17] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. "On Tight Security Proofs for Schnorr Signatures". In: *Advances in Cryptology – ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Kaoshiung, Taiwan, R.O.C.: Springer, Heidelberg, Germany, 2014, pp. 512–531. DOI: `10.1007/978-3-662-45611-8_27`.

[18] Eduarda S. V. Freire et al. "Non-Interactive Key Exchange". In: *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. Lecture Notes in Computer Science. Nara, Japan: Springer, Heidelberg, Germany, 2013, pp. 254–271. DOI: `10.1007/978-3-642-36362-7_17`.

[19] Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. "Improved Bounds on Security Reductions for Discrete Log Based Signatures". In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2008, pp. 93–107.

[20] Romain Gay, Dennis Hofheinz, and Lisa Kohl. "Kurosawa-Desmedt Meets Tight Security". In: *Advances in Cryptology – CRYPTO 2017, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2017, pp. 133–160.

[21] Romain Gay et al. "Tightly CCA-Secure Encryption Without Pairings". In: *Advances in Cryptology – EUROCRYPT 2016, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, 2016, pp. 1–27. DOI: `10.1007/978-3-662-49890-3_1`.

[22] Rosario Gennaro and Yehuda Lindell. "A Framework for Password-Based Authenticated Key Exchange". In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. `http://eprint.iacr.org/2003/032.ps.gz`. Warsaw, Poland: Springer, Heidelberg, Germany, 2003, pp. 524–543.

[23] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. “A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks”. In: *SIAM Journal on Computing* 17.2 (Apr. 1988), pp. 281–308.

[24] Junqing Gong et al. “Extended Nested Dual System Groups, Revisited”. In: *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I.* Ed. by Chen-Mou Cheng et al. Vol. 9614. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, Heidelberg, Germany, 2016, pp. 133–163. DOI: 10.1007/978-3-662-49384-7_6.

[25] Jens Groth, Rafail Ostrovsky, and Amit Sahai. “Non-interactive Zaps and New Techniques for NIZK”. In: *Advances in Cryptology – CRYPTO 2006.* Ed. by Cynthia Dwork. Vol. 4117. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2006, pp. 97–111.

[26] Fuchun Guo et al. “Optimal Security Reductions for Unique Signatures: Bypassing Impossibilities with a Counterexample”. In: *Advances in Cryptology – CRYPTO 2017, Part II.* Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2017, pp. 517–547.

[27] Dennis Hofheinz. “Adaptive Partitioning”. In: *Advances in Cryptology – EUROCRYPT 2017, Part II.* Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, 2017, pp. 489–518.

[28] Dennis Hofheinz. “Algebraic Partitioning: Fully Compact and (almost) Tightly Secure Cryptography”. In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I.* Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, 2016, pp. 251–281. DOI: 10.1007/978-3-662-49096-9_11.

[29] Dennis Hofheinz and Tibor Jager. “Tightly Secure Signatures and Public-Key Encryption”. In: *Advances in Cryptology – CRYPTO 2012.* Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2012, pp. 590–607.

[30] Dennis Hofheinz, Tibor Jager, and Edward Knapp. “Waters Signatures with Optimal Security Reduction”. In: *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography.* Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Vol. 7293. Lecture Notes in Computer Science. Darmstadt, Germany: Springer, Heidelberg, Germany, 2012, pp. 66–83.

[31] Dennis Hofheinz and Eike Kiltz. “Secure Hybrid Encryption from Weakened Key Encapsulation”. In: *Advances in Cryptology – CRYPTO 2007.* Ed. by Alfred Menezes. Vol. 4622. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2007, pp. 553–571.

[32] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. “Identity-Based Encryption with (Almost) Tight Security in the Multi-instance, Multi-ciphertext Setting”. In: *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography.* Ed. by Jonathan Katz. Vol. 9020. Lecture Notes in Computer Science. Gaithersburg, MD, USA: Springer, Heidelberg, Germany, 2015, pp. 799–822. DOI: 10.1007/978-3-662-46447-2_36.

[33] Saqib A. Kakvi and Eike Kiltz. “Optimal Security Proofs for Full Domain Hash, Revisited”. In: *Advances in Cryptology – EUROCRYPT 2012.* Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer, Heidelberg, Germany, 2012, pp. 537–553.

[34] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. “Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords”. In: *Advances in Cryptology – EUROCRYPT 2001.* Ed. by Birgit Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Innsbruck, Austria: Springer, Heidelberg, Germany, 2001, pp. 475–494.

[35] Jonathan Katz and Nan Wang. “Efficiency Improvements for Signature Schemes with Tight Security Reductions”. In: *ACM CCS 03: 10th Conference on Computer and Communications Security.* Ed. by Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger. Washington D.C., USA: ACM Press, 2003, pp. 155–164.

[36] Eike Kiltz and Hoeteck Wee. "Quasi-Adaptive NIZK for Linear Subspaces Revisited". In: *Advances in Cryptology – EUROCRYPT 2015, Part II.* Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Lecture Notes in Computer Science. Sofia, Bulgaria: Springer, Heidelberg, Germany, 2015, pp. 101–128. DOI: 10.1007/978-3-662-46803-6_4.

[37] Allison B. Lewko and Brent Waters. "Why Proving HIBE Systems Secure Is Difficult". In: *Advances in Cryptology – EUROCRYPT 2014.* Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Copenhagen, Denmark: Springer, Heidelberg, Germany, 2014, pp. 58–76. DOI: 10.1007/978-3-642-55220-5_4.

[38] Benoît Libert et al. "Compactly Hiding Linear Spans - Tightly Secure Constant-Size Simulation-Sound QA-NIZK Proofs and Applications". In: *Advances in Cryptology – ASIACRYPT 2015, Part I.* Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. Lecture Notes in Computer Science. Auckland, New Zealand: Springer, Heidelberg, Germany, 2015, pp. 681–707. DOI: 10.1007/978-3-662-48797-6_28.

[39] Benoît Libert et al. "Concise Multi-challenge CCA-Secure Encryption and Signatures with Almost Tight Security". In: *Advances in Cryptology – ASIACRYPT 2014, Part II.* Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. Lecture Notes in Computer Science. Kaoshiung, Taiwan, R.O.C.: Springer, Heidelberg, Germany, 2014, pp. 1–21. DOI: 10.1007/978-3-662-45608-8_1.

[40] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. "The Kernel Matrix Diffie-Hellman Assumption". In: *Advances in Cryptology – ASIACRYPT 2016, Part I.* Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. Lecture Notes in Computer Science. Hanoi, Vietnam: Springer, Heidelberg, Germany, 2016, pp. 729–758. DOI: 10.1007/978-3-662-53887-6_27.

[41] Moni Naor and Moti Yung. "Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks". In: *22nd Annual ACM Symposium on Theory of Computing.* Baltimore, MD, USA: ACM Press, 1990, pp. 427–437.

[42] Amit Sahai. "Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security". In: *40th Annual Symposium on Foundations of Computer Science.* New York, NY, USA: IEEE Computer Society Press, 1999, pp. 543–553.

[43] Yannick Seurin. "On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model". In: *Advances in Cryptology – EUROCRYPT 2012.* Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer, Heidelberg, Germany, 2012, pp. 554–571.