Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier

Michel Abdalla[1] Fabrice Benhamouda[1] Alain Passelègue[1] Kenneth G. Paterson[2]

[1] Département d'Informatique, École normale supérieure
45 Rue d'Ulm, 75230 Paris Cedex 05, France
{michel.abdalla,fabrice.ben.hamouda,alain.passelegue}@ens.fr
http://www.di.ens.fr/users/{mabdalla,fbenhamo,passeleg}
[2] Information Security Group, Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
kenny.paterson@rhul.ac.uk
http://www.isg.rhul.ac.uk/~kp/

Abstract

Related-key attacks (RKAs) concern the security of cryptographic primitives in the situation where the key can be manipulated by the adversary. In the RKA setting, the adversary's power is expressed through the class of related-key deriving (RKD) functions which the adversary is restricted to using when modifying keys. Bellare and Kohno (Eurocrypt 2003) first formalised RKAs and pin-pointed the foundational problem of constructing RKA-secure pseudorandom functions (RKA-PRFs). To date there are few constructions for RKA-PRFs under standard assumptions, and it is a major open problem to construct RKA-PRFs for larger classes of RKD functions. We make significant progress on this problem. We first show how to repair the Bellare-Cash framework for constructing RKA-PRFs and extend it to handle the more challenging case of classes of RKD functions that contain claws. We apply this extension to show that a variant of the Naor-Reingold function already considered by Bellare and Cash is an RKA-PRF for a class of affine RKD functions under the DDH assumption, albeit with an exponential-time security reduction. We then develop a second extension of the Bellare-Cash framework, and use it to show that the same Naor-Reingold variant is actually an RKA-PRF for a class of degree d polynomial RKD functions under the stronger decisional d-Diffie-Hellman inversion assumption. As a significant technical contribution, our proof of this result avoids the exponential-time security reduction that was inherent in the work of Bellare and Cash and in our first result.

---

1 Introduction

Background and Context. A common approach to prove the security of a cryptographic scheme, known as provable security, is to relate its security to one of its underlying primitives or to an accepted hard computational problem. While this approach is now standard and widely accepted, there is still a significant gap between the existing models used in security proofs and the actual environment in which these cryptosystems are deployed. For example, most of the existing security models assume that the adversary has no information about the user's secret key. However, it is well known that this is not always true in practice: the adversary may be able to learn partial information about the secrets using different types of side-channel attacks, such as the study of energy consumption, fault injection, or timing analysis. In the particular case of fault injection, for instance, an adversary can learn not only partial information about the secret key, but he may also be able to force a cryptosystem to work with different but related secret keys. Then, if he can observe the outcome of this cryptosystem, he may be able to break it. This is what is known in the literature as a related-key attack (RKA).

Most primitives are designed without taking related-key attacks into consideration so their security proofs do not provide any guarantee against such attacks. Hence, a cryptographic scheme that is perfectly safe in theory may be completely vulnerable in practice. Indeed, many such attacks were found during the last decade, especially against practical blockciphers [BDK05, BDK08, BDK+10, BK09, BKN09, KHP07]. Inspired by this cryptanalytic work, some years ago, theoreticians started to develop appropriate security models and search for cryptographic primitives which can be proven RKA secure.

Formal Foundations of RKA Security. Though RKAs were first introduced by Biham and Knudsen [Bih94, Knu93] in the early 1990s, it was only in 2003 that Bellare and Kohno [BK03] began the formalization of the theoretical foundations for RKA security. We recall their security definition for RKA security of PRFs here. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions for a security parameter κ, and let $\Phi = \{\phi\colon \mathcal{K} \to \mathcal{K}\}$ be a set of functions on the key space $\mathcal{K}$, called a related-key deriving (RKD) function set. We say that F is a Φ-RKA-PRF if for any polynomial-time adversary, its advantage in the following game is negligible. The game starts by picking a random challenge bit b, a random target key $K \in \mathcal{K}$ and a random function $G\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$. The adversary can repeatedly query an oracle that, given a pair $(\phi, x) \in \Phi \times \mathcal{D}$, returns either $F(\phi(K), x)$, if $b = 1$, or $G(\phi(K), x)$, if $b = 0$. Finally, the adversary outputs a bit b', and its advantage is defined by $2\Pr[\, b = b' \,] - 1$. Note that if the class Φ of RKD functions contains only the identity function, then this notion matches standard PRF security.

Bellare and Cash [BC10a] designed the first RKA-PRFs secure under standard assumptions, by adapting the Naor-Reingold PRF [NR97]. Their RKA-PRFs are secure for RKD function classes consisting of certain multiplicative and additive classes. To explain their results, let us begin by recalling the definition of the Naor-Reingold PRF. Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Let $\mathsf{NR}\colon (\mathbb{Z}_p^*)^{n+1} \times \{0,1\}^n \to \mathbb{G}$ denote the Naor-Reingold PRF that given a key $\vec{a} = (a_0, \ldots, a_n) \in (\mathbb{Z}_p^*)^{n+1}$ and input $x = x_1 \,\|\, \ldots \,\|\, x_n \in \{0,1\}^n$ returns

$$\mathsf{NR}(\vec{a}, x) = \left[a_0 \prod_{i=1}^{n} a_i^{x_i} \right]$$

where for any $a \in \mathbb{Z}_p$, $[a]$ stands for g^a, as defined in [EHK+13]. The key space of the Naor-Reingold PRF is $\mathcal{K} = (\mathbb{Z}_p^*)^{n+1}$, which has a group structure under the operation of component-wise multiplication modulo p, denoted $*$. Now let Φ_* denote the class of component-wise multiplicative functions on $(\mathbb{Z}_p^*)^{n+1}$, that is $\Phi_* = \{\phi\colon \vec{a} \in (\mathbb{Z}_p^*)^{n+1} \mapsto \vec{b} * \vec{a} \mid \vec{b} \in (\mathbb{Z}_p^*)^{n+1}\}$. It is easy to see

that NR is not itself a Φ_*-RKA-PRF, since it suffers from simple algebraic attacks, but using a collision-resistant hash function $h\colon \{0,1\}^n \times \mathbb{G}^{n+1} \to \{0,1\}^{n-2}$, Bellare and Cash were able to show that a simple modification of the Naor-Reingold PRF does yield a Φ_*-RKA-PRF under the DDH assumption. Specifically, they defined $F\colon (\mathbb{Z}_p^*)^{n+1} \times \{0,1\}^n \to \mathbb{G}$ by:

$$F(\vec{a}, x) = \mathsf{NR}(\vec{a}, 11 \,\|\, h(x, ([a_0], [a_0 a_1], \ldots, [a_0 a_n])))$$

and showed that this F is indeed a Φ_*-RKA-PRF under the DDH assumption. A second construction in [BC10a] uses similar techniques to build an RKA-PRF under the DLIN assumption.

In the original version of their paper, Bellare and Cash also used a variant of the Naor-Reingold PRF, $\mathsf{NR}^*\colon \mathbb{Z}_p^n \times \{0,1\}^n \setminus \{0^n\} \to \mathbb{G}$, defined by:

$$\mathsf{NR}^*(\vec{a}, x) = \left[\prod_{i=1}^{n} a_i^{x_i}\right],$$

to obtain a third RKA-PRF, this one for additive RKD functions. In more detail, the key space $\mathcal{K} = \mathbb{Z}_p^n$ of NR^*, has a natural group structure under the operation of component-wise addition modulo p. We define Φ_+ to be the class of functions, $\Phi_+ = \{\phi\colon \vec{a} \in \mathbb{Z}_p^n \mapsto \vec{a} + \vec{b} \mid \vec{b} \in \mathbb{Z}_p^n\}$. Then, Bellare and Cash claimed that the function $F\colon \mathbb{Z}_p^n \times \{0,1\}^n \setminus 0^n \to \mathbb{G}$ with

$$F(\vec{a}, x) = \mathsf{NR}^*(\vec{a}, 11 \,\|\, h(x, ([a_1], [a_2], \ldots, [a_n])))$$

is a Φ_+-RKA-PRF under the DDH assumption, when the function $h\colon \{0,1\}^n \times \mathbb{G}^n \to \{0,1\}^{n-2}$ is a collision-resistant hash function. The running time of their security reduction in this case was exponential in the input size.

These foundational results of [BC10a] were obtained by applying a single, elegant, general framework to the Naor-Reingold PRFs. The framework hinges on two main tools, key-malleability and key-fingerprints for PRFs and associated RKD function classes Φ. The former property means that there is an efficient deterministic algorithm, called a *key-transformer*, that enables one to transform an oracle for computing $M(K, x)$ into one for computing $M(\phi(K), x)$ for any $\phi \in \Phi$ and any input x (the technical requirements are in fact somewhat more involved than these), where M denotes the PRF on which one would like to apply the framework. The latter provides a means to ensure that, in the Bellare-Cash construction for an RKA-PRF from a (normal) PRF M, all adversarial queries to the putative Φ-RKA-PRF get appropriately separated before being processed by M. In combination, these two features enable a reduction to be made to the PRF security of the underlying function M.

Unfortunately, it was recently discovered that the original framework of [BC10a] has a bug, in that a technical requirement on the key-transformer, called hash function compatibility, was too weak to enable the original security proof of the Bellare-Cash construction to go through. When hash function compatibility is appropriately strengthened to enable a proof, it still holds for the key-transformers used in the analysis of their two main constructions, the multiplicative DDH and DLIN-based RKA-PRF constructions. However, the new compatibility definition no longer holds for the key-transformer used in their additive, DDH-based RKA-PRF construction. With respect to their framework and, specifically, their additive, DDH-based RKA-PRF construction, Bellare and Cash note in the latest version of their paper [BC10b]: *We see no easy way to fill the gap within our current framework and accordingly are retracting our claims about this construction and omitting it from the current version.*

Main Question. A natural question that arises from the work of Bellare-Cash is whether it is possible to go further, to obtain RKA-PRFs for larger classes of RKD function than Φ_* and Φ_+. This is important in understanding whether there are yet to be discovered fundamental barriers in

achieving RKA security for PRFs, as well as bringing the current state of the art for RKA security closer to practical application. This question becomes even more relevant in the light of the results of Bellare, Cash and Miller [BCM11], who showed that RKA-security can be transferred from PRFs to several other primitives, including identity-based encryption (IBE), signatures, as well as symmetric (SE) and public-key encryption (PKE) secure against chosen-ciphertext attacks. Their results illustrate the central role that RKA-PRFs play in related-key security more generally: any advance in constructing RKA-PRFs for broader classes would immediately transfer to these other primitives via the results of [BCM11]. A subsidiary question is whether it is possible to repair the Bellare-Cash framework without requiring stronger hash compatibility conditions on the key-transformer. This, if achievable, would reinstate their Φ_+-RKA-PRF.

A partial answer to the first question was provided by Goyal, O'Neill and Rao [GOR11], who proposed RKA secure weak-PRF and symmetric encryption schemes for polynomial functions using the Decisional Truncated q-ADHE problem. RKA secure weak-PRFs, however, are significantly weaker than standard RKA-PRFs since their security only holds with respect to random inputs. Wee [Wee12] provided RKA secure PKE for linear functions, while Bellare, Paterson, and Thomson [BPT12] proposed a framework for obtaining RKA secure IBE for affine and polynomial RKD function sets, from which RKA security for signatures, PKE (and more) for the same RKD function sets follows using the results of [BCM11] and extensions thereof. However, in respect of these works, it should be noted that achieving RKA security for randomized primitives appears to be substantially easier than for PRFs which are deterministic objects. An extended discussion on this point can be found in [BC10a, Section 1].

In parallel work to ours, Lewi et al. [LMR14] showed that the key homomorphic PRFs from Boneh et al. [BLMR13] (and slight extensions of them) are RKA secure. Specifically, they show RKA-security for a strict subset of Φ_+ for the PRF of [BLMR13] that is based on the Learning with Error (LWE) problem, and against a claw-free class of affine functions for the PRF of [BLMR13] that is based on multilinear maps. They also showed that, if the adversary's queries are restricted to unique inputs, these two PRFs are RKA secure for larger classes, namely a class of affine RKD functions (with a low-norm for the "linear" part) for the LWE-based PRF and a class of polynomial RKD functions for the PRF based on multilinear maps. These classes are not really comparable to our classes Φ_{aff} and Φ_d of affine and polynomial functions defined below, because the secret-key structures are slightly different. However, we remark that Lewi et al. [LMR14] do not deal with claw-free classes and do not show ways to leverage unique-input RKA security to full RKA security. We handle both of these issues in our paper, and it may be possible to extend our solutions to their setting. It should also be remarked that the construction of Barnahee and Peikert [BP14] may also yield another RKA secure PRF based on LWE.

Our Contributions. In this paper, we make substantial progress on the main question above, obtaining RKA-PRFs for substantially larger classes of RKD functions than were previously known. To ease notations, we consider our RKD functions to be vectors of multivariate polynomials so that each component is a multivariate polynomial in $\mathbb{Z}_p[T_1, \ldots, T_n] = \mathbb{Z}_p[\vec{T}]$, where $T_1, \ldots, T_n$ are unknowns. Along the way, we recover the original Bellare-Cash framework, showing that their original technical conditions on the key-transformer are in fact *already* sufficient to enable a (different) proof of RKA security to go through. Let us first introduce our main results on specific RKA-PRFs, and then explain the technical means by which they are obtained.

For p prime and $n, d \geq 1$, let Φ_d denote the class of functions from $\mathbb{Z}_p^n$ to $\mathbb{Z}_p^n$ each of whose component functions is a non-constant univariate polynomial of degree at most d. That is, we have:

$$\Phi_d = \left\{ \phi\colon\ \mathbb{Z}_p^n \to \mathbb{Z}_p^n \;\middle|\; \begin{array}{r} \vec{\phi} = (\phi_1, \ldots, \phi_n); \phi_i : \vec{T} \mapsto \sum_{j=0}^{d} \alpha_{i,j} \cdot T_i^j, \\ \forall i = 1, \ldots, n,\ (\alpha_{i,1}, \ldots, \alpha_{i,d}) \neq 0^d \end{array} \right\}.$$

For the special case $d = 1$, we denote Φ_1 by Φ_{aff} (aff for affine functions). Note that $\Phi_+ \subset \Phi_{\mathsf{aff}}$.

We will construct RKA-PRFs for the RKD function classes Φ_{aff} and Φ_d for each d. To this end, let $\mathbb{G} = \langle g \rangle$ be a group of prime order p, let $\overline{\mathcal{D}} = \{0,1\}^n \times \mathbb{G}^n$ and let $h\colon \overline{\mathcal{D}} \to \{0,1\}^{n-2}$ be a hash function. Let $\omega_i = 0^{i-1} \,\|\, 1 \,\|\, 0^{n-i}$, for $i = 1, \ldots, n$. Define $F\colon \mathbb{Z}_p^n \times (\{0,1\}^n \setminus 0^n) \to \mathbb{G}$ by:

$$F(\vec{a}, x) = \mathsf{NR}^*(\vec{a}, 11 \,\|\, h(x, \mathsf{NR}^*(\vec{a}, \vec{\omega})))$$

for all $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n$. This is the same F as in the withdrawn construction of [BC10a]. Theorems 4.5 and 6.11 show that this function is an RKA-PRF for both the RKD function classes Φ_{aff} and Φ_d (for each d), under reasonable hardness assumptions.

For our first result on the Φ_{aff}-RKA-PRF security of F, we recover and extend the withdrawn result of Bellare and Cash [BC10a], under the same hardness assumption that they required, namely the standard DDH assumption. Here our proof, like that in [BC10a], requires an exponential-time reduction. We then develop a further extension of the Bellare-Cash framework enabling us to circumvent their use of key-transformers having a key malleability property. We use this framework to modularize our proof that F is also a Φ_d-RKA-PRF. As part of this proof, we require the decisional d-Diffie-Hellman Inversion (d-DDHI) assumption, introduced in [GOR11]. Informally, the d-DDHI problem in a group $\mathbb{G}$ of prime order p consists of deciding, given inputs $([1], [a], \ldots, [a^d])$ and z, where $[1] = g$ is a generator of $\mathbb{G}$, whether z is equal to $\left[\frac{1}{a}\right]$ or to a random group element. Notably, in our analysis of the Φ_d-RKA-PRF security of F, we are able to avoid an exponential-time reduction. This puts the RKA-PRF F on the same footing as the surviving constructions in [BC10a].

Let us now expand on the technical aspects of our contributions.

Proof Barriers and Techniques. We first show how the Bellare-Cash framework can be modified to deal with RKD functions that are *not* claw-free, meaning that there exist pairs of different RKD functions ϕ_1 and ϕ_2 and a key $K \in \mathcal{K}$, such that $\phi_1(K) = \phi_2(K)$. Up to now, only claw-free classes have been considered for RKA-PRFs. But classes Φ underlying practical attacks such as fault injections have no reason to be claw-free, so dealing with non-claw-free classes of RKD functions is important in advancing RKA security towards practice. Moreover, both our RKD function classes of interest, Φ_{aff} and Φ_d, do contain claws. The lack of claw-freeness poses a problem in security proofs because, if an adversary is able to find two RKD functions which lead to the same derived key, he can detect this via his queries, and then the equation $\phi_1(K) = \phi_2(K)$ may leak information on K sufficient to enable the adversary to break RKA-PRF security in a particular construction.

We overcome the lack of claw-freeness in our adaptation of the Bellare-Cash framework by introducing two new concepts, Φ-Key-Collision Security for PRFs and Φ-Statistical-Key-Collision Security. The former is a property similar to the identity-collision-resistance property defined in [BCM11] in the context of pseudorandom generators and refers to the non-existence of an adversary who can find a colliding key (i.e. $\phi_1 \neq \phi_2$ s.t. $\phi_1(K) = \phi_2(K)$ for $\phi_1, \phi_2 \in \Phi$), when given oracle access to the PRF under related keys $\phi(K)$. The latter concept is essentially the same, but now oracle access to the PRF is replaced by oracle access to a random function. These properties are just the right ingredients necessary to generalize the Bellare-Cash framework to the non-claw-free case.

At the same time as dealing with claws, we are able to repair the gap in the proof for the original Bellare-Cash framework, showing that the original hash function compatibility condition required of the key-transformer is already strong enough to enable an alternative proof of RKA security. Our new proof introduces a slightly different sequence of game hops in order to avoid the apparent impasse in the original proof. Our main theorem establishing the RKA-PRF security of functions arising from this framework is Theorem 3.1. It repairs and extends the corresponding main theorem in [BC10a]. Our theorem is then combined with an analysis of the specific function NR^* to obtain Theorem 4.5 concerning the Φ_{aff}-RKA-PRF security of F.

To show that F is also a Φ_d-RKA-PRF, we still have a second major difficulty to overcome. While Φ_d-key-collision security and Φ_d-statistical-key-collision security can still be proven for F, we no longer have the key-transformer component that is critical to the Bellare-Cash framework. Instead, in Section 5, we introduce a further extension of their framework, replacing the key-transformer with a stronger pseudorandomness condition on the base PRF M used in the construction, which we call $(\mathcal{S}, \Phi)$-unique-input-prf-rka security. The new requirement essentially states that M should already act as a Φ-RKA-PRF on a restricted domain $\mathcal{S}$, provided the queries $(\phi_1, x_1), \ldots, (\phi_q, x_q)$ made by the Φ-RKA-PRF adversary to its oracle with $x_i \in \mathcal{S}$ are all for *distinct* x_i. Under this condition, we are able to prove Theorem 5.1 establishing the security of RKA-PRFs arising from our further extension of the Bellare-Cash framework. This theorem then enables us to prove in a modular fashion that F is also a Φ_d-RKA-PRF.

The final technical challenge is in proving that NR^*, playing the role of M, satisfies the relevant $(\mathcal{S}, \Phi)$-unique-input-prf-rka security property so as to allow the application of Theorem 5.1. This is done in a crucial lemma, Lemma 6.3, whose proof involves a delicate series of hybrids in which we gradually replace the oracle responses to queries (ϕ_i, x_i) for x_i in a suitable set $\mathcal{S}$ with random values. We exploit the algebraic nature of the function NR^* to ensure that the hybrids are close under a particular pair of hardness assumptions (the (N, d)-PDDH and (N, d)-EDDH assumptions, which are stated in the proof). We also make use of an efficient, approximate (but close to perfect) procedure to detect linear dependencies arising in the simulation from the adversary's oracle queries. This procedure is key to making the entire proof efficient (rather than exponential-time). Finally, we provide a series of reductions relating our pair of hardness assumptions to the d-DDHI assumption. Examining the details of the proof shows that we can recover our result concerning Φ_{aff}-RKA-PRF security of F under DDH (rather than DDHI), but now without an exponential-time reduction.

Publication Note. An extended abstract of this paper appears in the Proceedings of the 34th Annual Cryptology Conference (CRYPTO 2014), Part I, Juan A. Garay and Rosario Gennaro (Eds.), volume 8616 of Lecture Notes in Computer Science, pages 77–94, Springer, August 2014. This is the full version.

2 Definitions

Notations and Conventions. Let κ denote the security parameter. Let $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ be the set of all functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$. A family of functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ for a security parameter κ takes a key $K \in \mathcal{K}$ and an input $x \in \mathcal{D}$ and returns an output $F(K, x) \in \mathcal{R}$. If $\vec{x}$ is a vector then $|\vec{x}|$ denotes its length, and $\vec{x} = (x_1, \ldots, x_{|\vec{x}|})$. For a binary string x, we denote $|x|$ its length, x_i its i-th bit and, for $i, j \in \{1, \ldots, n\}$, $i \leq j$, $x_{i,\ldots,j}$ the binary string $x_i \,\|\, \ldots \,\|\, x_j$. For a binary string $x \in \{0,1\}^n$ and an integer d, we denote by $d \cdot x$ the string $y = y_1 \,\|\, \ldots \,\|\, y_n \in \{0, d\}^n$ defined by $y_i = d \cdot x_i$ for $i = 1, \ldots, n$. For two strings $x, y \in \{0, \ldots, d\}^n$, we denote by $y \preceq x$ the fact that $y_i \leq x_i$, for any $i = 1, \ldots, n$ and we denote by $S(x)$ the set $\{i \mid x_i \neq 0\}$. We denote by $\boldsymbol{A}$ a matrix of size $k \times m$ and by $A_{i,j}$ its coefficients, for $i, j \in \{1, \ldots, k\} \times \{1, \ldots, m\}$. If $\vec{\phi}$ is a vector of functions from S_1 to S_2 with $|\vec{\phi}| = n$ and $\vec{a} \in S_1^n$ then we denote by $\vec{\phi}(\vec{a})$ the vector $(\phi_1(\vec{a}_1), \ldots, \phi_n(\vec{a}_n)) \in S_2^n$. If $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is a family of functions and $\vec{x}$ is a vector over $\mathcal{D}$ then $F(K, \vec{x})$ denotes the vector $(F(K, x_1), \ldots, F(K, x_{|\vec{x}|}))$. If S is a set, then $|S|$ denotes its size. We denote by $s \xleftarrow{\$} S$ the operation of picking at random s in S. If $\mathscr{A}$ is a randomized algorithm, we denote by $y \xleftarrow{\$} \mathscr{A}(x_1, x_2, \ldots)$ the operation of running $\mathscr{A}$ on inputs $(x_1, x_2, \ldots)$ with fresh coins and letting y denote the output.

Finally, following [EHK+13], we often implicitly consider a multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order p and we denote by $[a]_g$, or simply $[a]$ if there is no ambiguity about the generator, the element g^a.

Games. Some of our definitions and proofs use code-based game-playing [BR06]. Recall that a game has an **Initialize** procedure, procedures to respond to adversary's oracle queries, and a **Finalize** procedure. A game G is executed with an adversary $\mathscr{A}$ as follows. First, **Initialize** executes and its outputs are the inputs to $\mathscr{A}$. Then $\mathscr{A}$ executes, its oracle queries being answered by the corresponding procedures of G. When $\mathscr{A}$ terminates, its outputs become the input to the **Finalize** procedure. The output of the latter, denoted $\mathrm{G}^{\mathscr{A}}$ is called the output of the game, and we let "$\mathrm{G}^{\mathscr{A}} \Rightarrow 1$", abbreviated Succ in the proofs, denote the event that this game output takes the value 1. Boolean flags are assumed initialized to false. Games G_i, G_j are identical until flag if their code differs only in statements that follow the setting of flag to true. The running time of an adversary by convention is the worst case time for the execution of the adversary with any of the games defining its security, so that the time of the called game procedures is included.

PRFs. PRFs were introduced by [GGM84]. A PRF is a family of functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ which is efficiently computable and so that it is hard to distinguish a function chosen randomly from the PRF family from a random function, which is formally defined as the fact that the advantage of any efficient adversary in attacking the standard prf security of F is negligible. The advantage of an adversary $\mathscr{A}$ in attacking the standard prf security of a family of functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is defined via

$$\mathbf{Adv}^{\mathsf{prf}}_{F}(\mathscr{A}) = \Pr\left[\mathrm{PRFReal}^{\mathscr{A}}_{F} \Rightarrow 1\right] - \Pr\left[\mathrm{PRFRand}^{\mathscr{A}}_{F} \Rightarrow 1\right].$$

Game $\mathrm{PRFReal}_F$ begins by picking $K \xleftarrow{\$} \mathcal{K}$ and responds to query $\mathbf{Fn}(x)$ via $F(K, x)$. Game $\mathrm{PRFRand}_F$ begins by picking $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$ and responds to oracle query $\mathbf{Fn}(x)$ via $f(x)$.

RKA-PRFs. We recall the definitions from [BK03]. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions and $\Phi \subseteq \mathsf{Fun}(\mathcal{K}, \mathcal{K})$. The members of Φ are called RKD (Related-Key Deriving) functions. An adversary is said to be Φ-restricted if its oracle queries (ϕ, x) satisfy $\phi \in \Phi$. The advantage of a Φ-restricted adversary $\mathscr{A}$ in attacking the prf-rka security of F is defined via

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{F,\Phi}(\mathscr{A}) = \Pr\left[\mathrm{RKPRFReal}^{\mathscr{A}}_{F} \Rightarrow 1\right] - \Pr\left[\mathrm{RKPRFRand}^{\mathscr{A}}_{F} \Rightarrow 1\right].$$

Game $\mathrm{RKPRFReal}_F$ begins by picking $K \xleftarrow{\$} \mathcal{K}$ and then responds to oracle query $\mathbf{RKFn}(\phi, x)$ via $F(\phi(K), x)$. Game $\mathrm{RKPRFRand}_F$ begins by picking $G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ and responds to oracle query $\mathbf{RKFn}(\phi, x)$ via $G(\phi(K), x)$. We say that F is a Φ-RKA-secure PRF if for any Φ-restricted, efficient adversary, its advantage in attacking the prf-rka security is negligible.

Strong Key Fingerprint. A strong key fingerprint is a tool used in proofs to detect whether a key arises more than once in a simulation, even if we do not have any information about the key itself. We recall the definition from [BC10a]. Suppose $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is a family of functions. Let $\vec{\omega}$ be a vector over $\mathcal{D}$ and let $n = |\vec{\omega}|$. We say that $\vec{\omega}$ is a *strong key fingerprint* for F if

$$(F(K, \omega_1), \ldots, F(K, \omega_n)) \neq (F(K', \omega_1), \ldots, F(K', \omega_n))$$

for all distinct $K, K' \in \mathcal{K}$.

Key-Malleability. As defined in [BC10a], let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions and Φ be a class of RKD functions. Suppose KT is a deterministic algorithm that, given an oracle $f\colon \mathcal{D} \to \mathcal{R}$ and inputs $(\phi, x) \in \Phi \times \mathcal{D}$, returns a point $\mathsf{KT}^{f}(\phi, x) \in \mathcal{R}$. KT is said to be a *key-transformer* for (F, Φ) if it satisfies the *correctness* and *uniformity* conditions. *Correctness* asks that $\mathsf{KT}^{F(K,\cdot)}(\phi, x) = F(\phi(K), x)$ for every $(\phi, K, x) \in \Phi \times \mathcal{K} \times \mathcal{D}$. Let us say that a Φ-restricted adversary is unique-input if, in its oracle queries $(\phi_1, x_1), \ldots, (\phi_q, x_q)$, the points $x_1, \ldots, x_q$ are always distinct. *Uniformity* requires that for any (even inefficient) Φ-restricted, unique-input adversary $\mathscr{U}$,

$$\Pr\left[\mathrm{KTReal}^{\mathscr{U}}_{\mathsf{KT}} \Rightarrow 1\right] = \Pr\left[\mathrm{KTRand}^{\mathscr{U}}_{\mathsf{KT}} \Rightarrow 1\right],$$

proc Initialize // d-SDL
$g \xleftarrow{\$} \mathbb{G}$; $a \xleftarrow{\$} \mathbb{Z}_p^*$
Return $(g, [a], \ldots, [a^d])$
proc Finalize(a') // d-SDL
Return $([a] = [a'])$

proc Initialize // d-DDHI-Real
$g \xleftarrow{\$} \mathbb{G}$
$a \xleftarrow{\$} \mathbb{Z}_p^*$
Return $(g, [a], \ldots, [a^d], [1/a])$
proc Finalize(b) // d-DDHI-Real
Return b

proc Initialize // d-DDHI-Rand
$g \xleftarrow{\$} \mathbb{G}$
$a \xleftarrow{\$} \mathbb{Z}_p^*$; $z \xleftarrow{\$} \mathbb{Z}_p^*$
Return $(g, [a], \ldots, [a^d], [z])$
proc Finalize(b) // d-DDHI-Rand
Return b

Figure 1: Games defining the d-SDL and d-DDHI problems in $\mathbb{G}$.

where game $\mathrm{KTReal}_{\mathsf{KT}}$ is initialized by picking $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$ and responds to query $\mathbf{KTFn}(\phi, x)$ via $\mathsf{KT}^f(\phi, x)$, while $\mathrm{KTRand}_{\mathsf{KT}}$ has no initialization and responds to oracle query $\mathbf{KTFn}(\phi, x)$ by returning a value $y \xleftarrow{\$} \mathcal{R}$ chosen uniformly at random in $\mathcal{R}$. If such a key-transformer exists, we say that F is a Φ-key-malleable PRF.

Compatible Hash Function. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions and Φ be a class of RKD functions, such that there is a key-transformer KT for (F, Φ). Let $\vec{\omega} \in \mathcal{D}^m$ and let $\overline{\mathcal{D}} = \mathcal{D} \times \mathcal{R}^m$. We denote by $\mathsf{Qrs}(\mathsf{KT}, F, \Phi, \vec{\omega})$ the set of all $w \in \mathcal{D}$ such that there exists $(f, \phi, i) \in \mathsf{Fun}(\mathcal{D}, \mathcal{R}) \times \Phi \times \{1, \ldots, m\}$ such that the computation of $\mathsf{KT}^f(\phi, \omega_i)$ makes oracle query w. Then, we say that a hash function $H\colon \overline{\mathcal{D}} \to S$ is *compatible* with $(\mathsf{KT}, F, \Phi, \vec{\omega})$, if $S = \mathcal{D} \setminus \mathsf{Qrs}(\mathsf{KT}, F, \Phi, \vec{\omega})$. Note that this definition is the same as that given in the original Bellare-Cash framework [BC10a] rather than the stronger one used in the authors' repaired version [BC10b].

CR hash functions. The advantage of $\mathscr{C}$ in attacking the collision-resistance security of $H\colon \mathcal{D} \to \mathcal{R}$ is

$$\mathbf{Adv}^{\mathsf{cr}}_{H}(\mathscr{C}) = \Pr\left[\, x \neq x' \text{ and } H(x) = H(x') \,\right]$$

where the probability is over $(x, x') \xleftarrow{\$} \mathscr{C}$.

Hardness Assumptions. Our proofs make use of the d-Strong Discrete Logarithm (d-SDL) and Decisional d-Diffie-Hellman Inversion (d-DDHI) problems given in [GOR11] and described in Figure 1. We define the advantage of an adversary $\mathscr{D}$ against the d-SDL problem in $\mathbb{G}$ as

$$\mathbf{Adv}^{d\text{-}\mathsf{sdl}}_{\mathbb{G}}(\mathscr{D}) = \Pr\left[\, d\text{-SDL}^{\mathscr{D}}_{\mathbb{G}} \Rightarrow \mathsf{true} \,\right]$$

where the probability is over the choices of $a \in \mathbb{Z}_p$, $g \in \mathbb{G}$, and the random coins used by the adversary. The advantage of an adversary $\mathscr{D}$ against the d-DDHI problem in $\mathbb{G}$ is defined to be

$$\mathbf{Adv}^{d\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{D}) = \Pr\left[\, d\text{-DDHI-Real}^{\mathscr{D}}_{\mathbb{G}} \Rightarrow 1 \,\right] - \Pr\left[\, d\text{-DDHI-Rand}^{\mathscr{D}}_{\mathbb{G}} \Rightarrow 1 \,\right]$$

where the probabilities are over the choices of $a, z \in \mathbb{Z}_p$, $g \in \mathbb{G}$, and the random coins used by the adversary.

We have two assumptions corresponding to the hardness of these problems, the d-SDL assumption and the d-DDHI assumption. Setting $d = 1$ in the d-SDL problem, we recover the usual definition of the DL problem in $\mathbb{G}$.

3 Repairing and Extending the Bellare-Cash Framework

Here, we give a method to deal with classes of RKD functions that are not claw-free, such as affine classes, by repairing and extending the general framework of Bellare and Cash from [BC10a]. Our approach still relies on key-malleability, meaning that it is not generally applicable since almost all

proc Initialize
$K \xleftarrow{\$} \mathcal{K}$

proc RKFn(ϕ, x)
$y \leftarrow M(\phi(K), x)$
Return y

proc Finalize(ϕ_1, ϕ_2)
Return $(\phi_1 \neq \phi_2$ and $\phi_1(K) = \phi_2(K))$

proc Initialize
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$
$F \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$; $b' \leftarrow 0$

proc RKFn(ϕ, x)
If $\phi(K) \in E$ and $\phi \notin D$ then $b' \leftarrow 1$
$D \leftarrow D \cup \{\phi\}$; $E \leftarrow E \cup \{\phi(K)\}$
$y \leftarrow F(\phi(K), x)$
Return y

proc Finalize
Return $(b' = 1)$

Figure 2: Game defining the Φ-key-collision security of a PRF M on the left and Φ-statistical-key-collision security for $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ on the right.

the known PRFs are not key-malleable for interesting classes of functions. However, as we shall see, it *does* provide an easy way to obtain a Φ_{aff}-RKA-secure PRF, using the variant NR^* of the Naor-Reingold PRF. In Section 5, we will present a further extension of the Bellare-Cash approach that enables us to deal with PRFs that are not key-malleable.

To deal with non-claw-freeness, we first introduce two new notions. The first one is called Φ-*Key-Collision Security* and captures the likelihood that an adversary finds two RKD functions which lead to the same derived key in a given PRF construction. The second one, called Φ-*Statistical-Key-Collision Security*, is similar, but replaces the oracle access to the PRF with an oracle access to a random function.

Φ-Key-Collision (Φ-kc) Security. Let Φ be a class of RKD functions. We define the advantage of an adversary $\mathscr{A}$ against the Φ-key-collision security of a PRF $M\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$, denoted by $\mathbf{Adv}^{\mathsf{kc}}_{\Phi,M}(\mathscr{A})$, to be the probability of success in the game on the left side of Figure 2, where the functions ϕ appearing in $\mathscr{A}$'s queries are restricted to lie in Φ.

Φ-Statistical-Key-Collision (Φ-skc) Security. Let Φ be a class of RKD functions. We define the advantage of an adversary $\mathscr{A}$ against the Φ-statistical-key-collision security for $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$, denoted by $\mathbf{Adv}^{\mathsf{skc}}_{\Phi}(\mathscr{A})$, to be the probability of success in the game on the right side of Figure 2. Here the functions ϕ appearing in $\mathscr{A}$'s queries are again restricted to lie in Φ.

Using these notions, we can now prove the following theorem, which both repairs and extends the main result of [BC10a].

Theorem 3.1. *Let $M\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions and Φ be a class of* RKD *functions that contains the identity function* id. *Let KT be a key-transformer for (M, Φ) making Q_{KT} oracle queries, and let $\vec{\omega} \in \mathcal{D}^m$ be a strong key fingerprint for M. Let $\overline{\mathcal{D}} = \mathcal{D} \times \mathcal{R}^m$ and let $H\colon \overline{\mathcal{D}} \to S$ be a hash function that is compatible with $(\mathsf{KT}, M, \Phi, \vec{\omega})$. Define $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ by*

$$F(K, x) = M(K, H(x, M(K, \vec{\omega})))$$

for all $K \in \mathcal{K}$ and $x \in \mathcal{D}$. Let $\mathscr{A}$ be a Φ-restricted adversary against the prf-rka security of F that makes $Q_{\mathscr{A}} \leq |S|$ oracle queries. Then we can construct an adversary $\mathscr{B}$ against the standard prf security of M, an adversary $\mathscr{C}$ against the cr security of H, an adversary $\mathscr{D}$ against the Φ-kc security of M and an adversary $\mathscr{E}$ against Φ-skc security for $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ such that

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi,F}(\mathscr{A}) \;\leq\; \mathbf{Adv}^{\mathsf{prf}}_{M}(\mathscr{A}) + \mathbf{Adv}^{\mathsf{cr}}_{H}(\mathscr{C}) + \mathbf{Adv}^{\mathsf{kc}}_{\Phi,M}(\mathscr{D}) + \mathbf{Adv}^{\mathsf{skc}}_{\Phi}(\mathscr{E}). \tag{1}$$

Adversaries $\mathscr{C}$, $\mathscr{D}$ and $\mathscr{E}$ have the same running time as $\mathscr{A}$. Adversary $\mathscr{B}$ has the same running time as $\mathscr{A}$ plus the time required for $Q_{\mathscr{A}} \cdot (m + 1)$ executions of the key-transformer KT.

Note that if the class Φ is claw-free, then the advantage of any adversary in breaking Φ-kc security of M or Φ-skc security for $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ is zero. In this case Theorem 3.1 matches exactly the main theorem of [BC10a], under the original and weaker definition of hash function compatibility from [BC10a]. This justifies our claim of repairing the Bellare-Cash framework.

Overview of the Proof. The proof of the above theorem is detailed below and relies on the sequence of 11 games (games $\mathrm{G}_0 - \mathrm{G}_{10}$) described in Figure 3. Here we provide a brief overview. Since the RKD functions that we consider in our case may have claws, we start by dealing with possible collisions on the related-keys in the RKPRFReal case, using the key-collision notion (games $\mathrm{G}_0 - \mathrm{G}_2$). Then, in games $\mathrm{G}_3 - \mathrm{G}_4$, we deal with possible collisions on hash values in order to ensure that the hash values h used to compute the output y are pairwise distinct so the attacker is unique-input. Then, using the properties of the key-transformer and the compatibility condition, we show that it is hard to distinguish the output from a uniformly random output (games $\mathrm{G}_5 - \mathrm{G}_7$) based on the standard prf security of M. Finally, we use the statistical-key-collision security notion to deal with possible key collisions in the RKPRFRand case (games $\mathrm{G}_8 - \mathrm{G}_{10}$) so that G_{10} matches the description of the RKPRFRand game.

Remark 3.2. It is worth noting that we deviate from the original proof of [BC10a] in games $\mathrm{G}_5 - \mathrm{G}_7$, filling the gap in their original proof, but under the same technical conditions on compatibility. Unlike in their proof, we are able to show that the output of F is already indistinguishable from a uniformly random output as soon as one replaces the underlying PRF M with a random function f due to the uniformity condition of the transformer. In order to build a unique-input adversary against the uniformity condition, the main trick is to precompute the values of $f(w)$ for all $w \in \mathsf{Qrs}(\mathsf{KT}, M, \Phi, \vec{\omega})$ and use these values to compute $\mathsf{KT}^f(\phi, \omega_i)$, for $i = 1, \ldots, |\vec{\omega}|$ and $\phi \in \Phi$, whenever needed. This avoids the need to query the oracle in the uniformity game twice on the same input when computing the fingerprint.

Proof of Theorem 3.1. The proof is based on the sequence of games in Figure 3. Much of the proof is similar to that of the general framework of Bellare and Cash from [BC10a]. However, we have additional games to deal with non-claw-freeness (games G_1, G_2, G_9 and G_{10}), and some games (games G_6 and G_7) are modified to deal with the gap in the proof of the corresponding theorem in [BC10a]. Let $\textsc{Succ}_i$ denote the event that game G_i output takes the value 1.

Game G_1 introduces storage of used RKD-functions and values of $\vec{\omega}$ in sets D and E respectively and sets flag_1 to true if the same value of $\vec{\omega}$ arises for two different RKD-functions. Since this storage does not affect the values returned by **RKFn**

$$\Pr[\,\textsc{Succ}_1\,] = \Pr[\,\textsc{Succ}_0\,].$$

Game G_2 adds the boxed code which changes how the repetition of an $\vec{\omega}$ value is handled, by picking instead a random value from $\mathcal{R}^m \setminus E$ that will not repeat any previous one. Games G_1 and G_2 are identical until flag_1 is set to true, hence we have

$$\Pr[\,\textsc{Succ}_1\,] \;\leq\; \Pr[\,\textsc{Succ}_2\,] + \Pr[\,E_1\,]$$

where E_1 denotes the event that the execution of $\mathscr{A}$ with game G_1 sets flag_1 to true. We design an adversary $\mathscr{D}$ attacking the Φ-key-collision security of M such that

$$\Pr[\,E_1\,] \;\leq\; \mathbf{Adv}^{\mathsf{kc}}_{\Phi,M}(\mathscr{D}).$$

Adversary $\mathscr{D}$ runs $\mathscr{A}$. When the latter makes a **RKFn**-query (ϕ, x), adversary $\mathscr{D}$ queries (ϕ, ω_i), for $i = 1, \ldots, |\vec{\omega}|$, to its oracle, then computes $\vec{\omega}$ and then $h = H(x, \vec{\omega})$ and finally queries (ϕ, h) to

proc Initialize // G_0
$K \xleftarrow{\$} \mathcal{K}$

proc RKFn(ϕ, x) // G_0
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\overline{\omega}_i \leftarrow M(\phi(K), \omega_i)$
$h \leftarrow H(x, \vec{\overline{\omega}})$
$y \leftarrow M(\phi(K), h)$
Return y

proc Finalize(b') // All Games
Return b'

---

proc Initialize // $\mathrm{G}_1, \mathrm{G}_2$
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$

proc RKFn(ϕ, x) // G_1, $\boxed{\mathrm{G}_2}$
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\overline{\omega}_i \leftarrow M(\phi(K), \omega_i)$
If $\vec{\overline{\omega}} \in E$ and $\phi \notin D$ then
  $\mathsf{flag}_1 \leftarrow \mathsf{true}$; $\boxed{\vec{\overline{\omega}} \xleftarrow{\$} \mathcal{R}^m \setminus E}$
$\boxed{\text{Else}}$ $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\overline{\omega}}\}$
$h \leftarrow H(x, \vec{\overline{\omega}})$
$y \leftarrow M(\phi(K), h)$
Return y

---

proc Initialize // $\mathrm{G}_3, \mathrm{G}_4$
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$; $G \leftarrow \emptyset$

proc RKFn(ϕ, x) // G_3, $\boxed{\mathrm{G}_4}$
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\overline{\omega}_i \leftarrow M(\phi(K), \omega_i)$
If $\vec{\overline{\omega}} \in E$ and $\phi \notin D$ then
  $\vec{\overline{\omega}} \xleftarrow{\$} \mathcal{R}^m \setminus E$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\overline{\omega}}\}$
$h \leftarrow H(x, \vec{\overline{\omega}})$
If $h \in G$ then $\mathsf{flag}_2 \leftarrow \mathsf{true}$
  $\boxed{h \xleftarrow{\$} \mathcal{S} \setminus G}$
$G \leftarrow G \cup \{r\}$
$y \leftarrow M(\phi(K), h)$
Return y

---

proc Initialize // G_5
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$; $G \leftarrow \emptyset$

proc RKFn(ϕ, x) // G_5
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\overline{\omega}_i \leftarrow \mathsf{KT}^{M(K,\cdot)}(\phi, \omega_i)$
If $\vec{\overline{\omega}} \in E$ and $\phi \notin D$ then
  $\vec{\overline{\omega}} \xleftarrow{\$} \mathcal{R}^m \setminus E$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\overline{\omega}}\}$
$h \leftarrow H(x, \vec{\overline{\omega}})$
If $h \in G$ then $h \xleftarrow{\$} \mathcal{S} \setminus G$
$G \leftarrow G \cup \{r\}$
$y \leftarrow \mathsf{KT}^{M(K,\cdot)}(\phi, h)$
Return y

---

proc Initialize // G_6
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$; $G \leftarrow \emptyset$
$f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$

proc RKFn(ϕ, x) // G_6
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\overline{\omega}_i \leftarrow \mathsf{KT}^f(\phi, \omega_i)$
If $\vec{\overline{\omega}} \in E$ and $\phi \notin D$ then
  $\vec{\overline{\omega}} \xleftarrow{\$} \mathcal{R}^m \setminus E$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\overline{\omega}}\}$
$h \leftarrow H(x, \vec{\overline{\omega}})$
If $h \in G$ then $h \xleftarrow{\$} \mathcal{S} \setminus G$
$G \leftarrow G \cup \{r\}$
$y \leftarrow \mathsf{KT}^f(\phi, h)$
Return y

---

proc Initialize // $\boxed{\mathrm{G}_8}$, G_9
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$
$G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$

proc RKFn(ϕ, x) // $\boxed{\mathrm{G}_8}$, G_9
$y \leftarrow G(\phi(K), x)$
If $\phi(K) \in E$ and $\phi \notin D$ then
  $\mathsf{flag}_3 \leftarrow \mathsf{true}$; $\boxed{y \xleftarrow{\$} \mathcal{R}}$
$D \leftarrow D \cup \{\phi\}$; $E \leftarrow E \cup \{\phi(K)\}$
Return y

---

proc Initialize // G_7
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$
$G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$

proc RKFn(ϕ, x) // G_7
$y \xleftarrow{\$} \mathcal{R}$
Return y

---

proc Initialize // G_{10}
$K \xleftarrow{\$} \mathcal{K}$; $G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$

proc RKFn(ϕ, x) // G_{10}
$y \leftarrow G(\phi(K), x)$
Return y

Figure 3: Games for the proof of Theorem 3.1.

its oracle and sends it to $\mathscr{A}$. When $\mathscr{A}$ stops, $\mathscr{D}$ searches for two different RKD-functions ϕ queried by $\mathscr{A}$ that lead to the same value $\vec{\omega}$ and returns these two functions if found. Since $\vec{\omega}$ is a strong key fingerprint, two such functions lead to the same key, so $\mathscr{D}$ wins if he finds such two functions. (Of course, if the class of RKD-functions is claw-free, the advantage of the attacker is 0.)

Game G_3 introduces storage of hash values in a set G and sets flag_2 to true if the same hash output arises twice. Since this storage does not affect the values returned by **RKFn**

$$\Pr[\,\textsc{Succ}_3\,] = \Pr[\,\textsc{Succ}_2\,].$$

Game G_4 adds the boxed code which changes how repetition of hash values is handled, by picking instead a random value h from $\mathcal{S} \setminus G$ that will not repeat any previously used hash value. Games G_3 and G_4 are identical until flag_2 is set to true, hence we have

$$\Pr[\,\textsc{Succ}_3\,] \;\le\; \Pr[\,\textsc{Succ}_4\,] + \Pr[\,E_2\,]$$

where E_2 denotes the event that the execution of $\mathscr{A}$ with game G_3 sets flag_2 to true. We design an adversary $\mathscr{C}$ attacking the cr security of H such that

$$\Pr[\,E_2\,] \;\le\; \mathbf{Adv}^{\mathsf{cr}}_H(\mathscr{C}).$$

Adversary $\mathscr{C}$ starts by picking $K \xleftarrow{\$} \mathcal{K}$ and initializes $j \leftarrow 0$. It runs $\mathscr{A}$. When the latter makes an **RKFn**-query (ϕ, x), adversary $\mathscr{C}$ responds via:

For $i = 0, \ldots, |\vec{\omega}|$ do: $\overline{\omega}_i \leftarrow M(\phi(K), \omega_i)$
$j \leftarrow j + 1$; $\phi_j \leftarrow \phi$; $x_j \leftarrow x$
If $\vec{\omega} \in E$ and $\phi \notin D$ then $\vec{\omega} \xleftarrow{\$} \mathcal{S} \setminus E$ $\qquad (*)$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\omega}\}$
$w_j \leftarrow \vec{\omega}$
$h \leftarrow H(x, \vec{\omega})$
$h_j \leftarrow h$
$y \leftarrow M(\phi(K), h)$
Return y.

When $\mathscr{A}$ halts, $\mathscr{C}$ searches for a, b satisfying $1 \le a < b \le j$ such that $h_a = h_b$ and, if it finds them, outputs $(x_a, w_a), (x_b, w_b)$ and halts. The pairs (x_a, w_a) and (x_b, w_b) are distinct. Indeed, consider two cases: first, if $\phi_a = \phi_b$ then since $\mathscr{A}$ never repeats an oracle query, $x_a \neq x_b$ hence $(x_a, w_a) \neq (x_b, w_b)$. Second, if $\phi_a \neq \phi_b$, then condition $(*)$ ensures that $w_a \neq w_b$. Hence once again, $(x_a, w_a) \neq (x_b, w_b)$, and then

$$\Pr[\,\textsc{Succ}_3\,] \;\le\; \Pr[\,\textsc{Succ}_4\,] + \mathbf{Adv}^{\mathsf{cr}}_H(\mathscr{C}).$$

In game G_5, we use the key transformer KT to compute $M(\phi(K), \cdot)$ via oracle calls to $M(K, \cdot)$. The correctness property of the key transformer implies

$$\Pr[\,\textsc{Succ}_4\,] = \Pr[\,\textsc{Succ}_5\,].$$

In game G_6, we replace the oracle $M(K, \cdot)$ given to the key transformer KT by a random function f. We design an adversary $\mathscr{B}$ attacking the prf security of M such that

$$\Pr[\,\textsc{Succ}_5\,] \;\le\; \Pr[\,\textsc{Succ}_6\,] + \mathbf{Adv}^{\mathsf{prf}}_M(\mathscr{B}).$$

Adversary $\mathscr{B}$ runs $\mathscr{A}$. When the latter makes an RKFn-query (ϕ, x), adversary $\mathscr{B}$ responds via

For $i = 0, \ldots, |\vec{\omega}|$ do $\overline{\omega}_i \leftarrow \mathsf{KT}^{\mathbf{Fn}}(\phi, \omega_i)$
If $\vec{\overline{\omega}} \in E$ and $\phi \notin D$ then $\vec{\overline{\omega}} \xleftarrow{\$} \mathcal{S} \setminus E$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\overline{\omega}}\}$
$h \leftarrow H(x, \vec{\overline{\omega}})$
$y \leftarrow \mathsf{KT}^{\mathbf{Fn}}(\phi, h)$
Return y

where **Fn** is $\mathscr{B}$'s own oracle. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. Then

$$\Pr\left[\,\mathrm{PRFReal}_M^{\mathscr{B}} \Rightarrow 1\,\right] = \Pr\left[\,\textsc{Succ}_5\,\right] \quad \text{and} \quad \Pr\left[\,\mathrm{PRFRand}_M^{\mathscr{B}} \Rightarrow 1\,\right] = \Pr\left[\,\textsc{Succ}_6\,\right].$$

In game G_7, instead of computing the output y using the key-transformer, we set the value y to a uniformly random value. To show that games G_6 and G_7 are perfectly indistinguishable, we use the uniformity condition of the Key-Transformer KT. Let us recall that, as formally defined in [BC10a, Section 3.1], the uniformity condition states that for any (even inefficient) Φ-restricted, unique-input adversary $\mathscr{U}$,

$$\Pr\left[\,\mathrm{KTReal}_{\mathsf{KT}}^{\mathscr{U}} \Rightarrow 1\,\right] = \Pr\left[\,\mathrm{KTRand}_{\mathsf{KT}}^{\mathscr{U}} \Rightarrow 1\,\right],$$

where game $\mathrm{KTReal}_{\mathsf{KT}}$ picks $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$ during the initialization and responds to oracle query $\mathbf{KTFn}(\phi, x)$ via $\mathsf{KT}^f(\phi, x)$, while game $\mathrm{KTRand}_{\mathsf{KT}}$ has no initialization and responds to oracle query $\mathbf{KTFn}(\phi, x)$ by returning a value $y \xleftarrow{\$} \mathcal{R}$ chosen uniformly at random in $\mathcal{R}$. We show that if an adversary $\mathscr{A}$ can distinguish games G_6 and G_7, then we can construct a unique-input adversary $\mathscr{U}$ that can distinguish games $\mathrm{KTReal}_{\mathsf{KT}}$ and $\mathrm{KTRand}_{\mathsf{KT}}$; since KT is a key-transformer, these two games are perfectly indistinguishable for a unique-input adversary by the uniformity condition. Hence, so are G_6 and G_7.

Adversary $\mathscr{U}$ starts by initializing sets $D \leftarrow \emptyset$, $E \leftarrow \emptyset$, $G \leftarrow \emptyset$, then makes the queries (id, w) to its oracle, for every $w \in \mathsf{Qrs}(\mathsf{KT}, M, \Phi, \vec{\omega})$ and stores these values. This is possible under our assumption that $\mathsf{id} \in \Phi$. We let f_w denote the value that $\mathscr{D}$ gets from its oracle in response to the query (id, w). Depending on $\mathscr{U}$'s oracle, the value of f_w for $w \in \mathsf{Qrs}(\mathsf{KT}, M, \Phi, \vec{\omega})$ is either $\mathsf{KT}^f(\mathsf{id}, w) = f(w)$ ($\mathrm{KTReal}_{\mathsf{KT}}$), with f the random function defined in the **Initialize** procedure of $\mathrm{KTReal}_{\mathsf{KT}}$, or a uniformly random value from $\mathcal{R}$ ($\mathrm{KTRand}_{\mathsf{KT}}$). All these values will be used by $\mathscr{U}$ to compute the value $\vec{\overline{\omega}}$ in its simulation. Now, $\mathscr{U}$ runs $\mathscr{A}$. When $\mathscr{A}$ makes an oracle query (ϕ, x), $\mathscr{U}$ starts by computing the values $\overline{\omega}_i$, for $i = 1, \ldots, |\vec{\omega}|$, using the values f_w he has stored, the function ϕ he gets from $\mathscr{A}$, and the key-transformer KT. Note that, because $\mathscr{U}$ already queried (id, w) to its oracle for every $w \in \mathsf{Qrs}(\mathsf{KT}, M, \Phi, \vec{\omega})$, $\mathscr{U}$ is able to compute by itself the values $\overline{\omega}_i$, for $i = 1, \ldots, |\vec{\omega}|$. This is because, in making these queries, $\mathscr{U}$ already sets a value f_w for every $w \in \mathsf{Qrs}(\mathsf{KT}, M, \Phi, \vec{\omega})$, and this is the set of all values that might be needed in computing $\mathsf{KT}^f(\phi, \omega_i)$, for $i = 1, \ldots, |\vec{\omega}|$ and $\phi \in \Phi$. Notice that, in making these queries all at once at the beginning, $\mathscr{U}$ remains a unique-input adversary. After computing $\vec{\overline{\omega}}$, $\mathscr{U}$ checks if $\vec{\overline{\omega}} \in E$ and $\phi \notin D$. If these conditions hold, $\mathscr{U}$ picks $\vec{\overline{\omega}} \xleftarrow{\$} \mathcal{R}^m \setminus E$ at random, otherwise $\mathscr{U}$ sets $D \leftarrow D \cup \{\phi\}$. It then sets $E \leftarrow E \cup \{\vec{\overline{\omega}}\}$. Next, $\mathscr{U}$ computes $h \leftarrow H(x, \vec{\overline{\omega}})$ and checks if $h \in G$. If this holds, $\mathscr{U}$ picks $h \xleftarrow{\$} \mathcal{S} \setminus G$ at random. Notice that this step guarantees that all values h are in $\mathcal{S}$ and are all distinct as long as $\mathscr{A}$ makes at most $|\mathcal{S}|$ queries. Finally, $\mathscr{U}$ sets $G \leftarrow G \cup \{h\}$, makes the query (ϕ, h) to its oracle, and returns the value it gets, which is either $\mathsf{KT}^f(\phi, h)$ or a uniformly random value, to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{U}$ halts with the same output. The compatibility condition ensures that $\mathcal{S}$ does not contain any w with $w \in \mathsf{Qrs}(\mathsf{KT}, M, \Phi, \vec{\omega})$. It follows from these observations that $\mathscr{U}$ is a unique-input adversary. Finally, it is clear that if $\mathscr{U}$'s oracle is $\mathrm{KTReal}_{\mathsf{KT}}$, then it simulates exactly game G_6 with f being

the function f chosen at random in the **Initialize** procedure of game $\mathrm{KTReal}_{\mathsf{KT}}$. If $\mathscr{U}$'s oracle is $\mathrm{KTRand}_{\mathsf{KT}}$, then it simulates exactly game G_7 since the values given to $\mathscr{A}$ are uniformly random values. Then, we have

$$\Pr\left[\mathrm{KTReal}_{\mathsf{KT}}^{\mathscr{U}} \Rightarrow 1\right] = \Pr\left[\mathrm{SUCC}_6\right] \quad \text{and} \quad \Pr\left[\mathrm{KTRand}_{\mathsf{KT}}^{\mathscr{U}} \Rightarrow 1\right] = \Pr\left[\mathrm{SUCC}_7\right]$$

and then, since $\Pr\left[\mathrm{KTReal}_{\mathsf{KT}}^{\mathscr{U}} \Rightarrow 1\right] = \Pr\left[\mathrm{KTRand}_{\mathsf{KT}}^{\mathscr{U}} \Rightarrow 1\right]$ for any unique-input adversary $\mathscr{U}$, by the uniformity condition, we finally have

$$\Pr\left[\mathrm{SUCC}_6\right] = \Pr\left[\mathrm{SUCC}_7\right].$$

Games G_7 and G_8 are identical since even if two different queries lead to the same key, the "If" test ensures that the returned value is still uniformly random over $\mathcal{R}$. Hence,

$$\Pr\left[\mathrm{SUCC}_7\right] = \Pr\left[\mathrm{SUCC}_8\right].$$

Games G_8 and G_9 are identical until flag_3 is set to true, hence we have

$$\Pr\left[\mathrm{SUCC}_8\right] \;\leq\; \Pr\left[\mathrm{SUCC}_9\right] + \Pr\left[E_3\right]$$

where E_3 denotes the event that the execution of $\mathscr{A}$ with game G_9 sets flag_3 to true. We design an adversary $\mathscr{E}$ breaking Φ-skc security for $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ such that:

$$\Pr\left[E_3\right] \;\leq\; \mathbf{Adv}^{\mathsf{skc}}_{\Phi}(\mathscr{E}).$$

Adversary $\mathscr{E}$ runs $\mathscr{A}$. When the latter makes a **RKFn**-query (ϕ, x), so does $\mathscr{E}$ and $\mathscr{E}$ returns the value he receives to $\mathscr{A}$. When $\mathscr{A}$ stops, if $\mathscr{A}$ has queried two different functions ϕ_1 and ϕ_2 such that $\phi_1(K) = \phi_2(K)$ then b' was set to 1 when the second of these two functions was queried by $\mathscr{E}$, and then $\mathscr{E}$ wins. (Of course, if the class of RKD functions is claw-free, this probability is 0.)

Games G_9 and G_{10} are identical, so

$$\Pr\left[\mathrm{SUCC}_9\right] = \Pr\left[\mathrm{SUCC}_{10}\right].$$

Equation (1) on page 8 now follows by combining the bounds arising in the different game hops. □

4 Related-Key Security for Affine RKD Functions

In this section, we apply the above framework to the variant NR^* of the Naor-Reingold PRF. Recall that NR^*: $\mathbb{Z}_p^n \times \{0,1\}^n \setminus \{0^n\} \to \mathbb{G}$ was defined in [BC10a] by:

$$\mathsf{NR}^*(\vec{a}, x) = \left[\prod_{i=1}^{n} a_i^{x_i}\right]$$

for all $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. We recall the definition of Φ_{aff} $(= \Phi_1)$ from the introduction. Using the above theorem, we prove that NR^* can be used to build a Φ_{aff}-RKA-secure PRF under the DDH assumption, thereby recovering and strengthening the withdrawn result from [BC10a]. We first recall the following lemma from [BC10a].

Lemma 4.1. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let NR^* be defined via $\mathsf{NR}^*(\vec{a}, x) = [\prod_{i=1}^n a_i^{x_i}]$, where $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. Let $\mathscr{A}$ be an adversary against the standard prf security of NR^* that makes $Q_{\mathscr{A}}$ oracle queries. Then we can construct an adversary $\mathscr{B}$ against the* DDH *problem such that*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{NR}^*}(\mathscr{A}) \;\leq\; (n-1) \cdot \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathscr{B}).$$

The running time of B is equal to the running time of A, plus the time required to compute $O(Q_{\mathscr{A}})$ exponentiations in $\mathbb{G}$.

In what follows, we prove the properties needed to apply Theorem 3.1 to NR^*.

Strong Key Fingerprint. Let $\omega_i = 0^{i-1} \,\|\, 1 \,\|\, 0^{n-i}$, for $i = 1, \ldots, n$. Then $\vec{\omega}$ is a strong key fingerprint for NR^*. Indeed, we have $(\mathsf{NR}^*(\vec{a}, \omega_1), \ldots, \mathsf{NR}^*(\vec{a}, \omega_n)) = ([a_1], \ldots, [a_n])$, so if $\vec{a} \neq \vec{a}'$ are two distinct keys in $\mathcal{K} = \mathbb{Z}_p^n$, then there exists $i \in \{1, \ldots, n\}$ such that $a_i \neq a'_i$, so $[a_i] \neq [\vec{a}'_i]$.

Compatible Hash Function. We have $\mathsf{Qrs}(\mathsf{KT}_{\Phi_{\mathsf{aff}}}, \mathsf{NR}^*, \Phi_{\mathsf{aff}}, \vec{\omega}) = \{\omega_1, \ldots, \omega_n\}$, so let $\overline{\mathcal{D}} = \{0,1\}^n \times \mathbb{G}^n$ and let $h\colon \overline{\mathcal{D}} \to \{0,1\}^{n-2}$ be a collision resistant hash function. Then the hash function defined by $H(x, \vec{z}) = 11 \,\|\, h(x, \vec{z})$ is a collision resistant hash function that is compatible with $(\mathsf{KT}_{\Phi_{\mathsf{aff}}}, \mathsf{NR}^*, \Phi_{\mathsf{aff}}, \vec{\omega})$ since every element of $\mathsf{Qrs}(\mathsf{KT}_{\Phi_{\mathsf{aff}}}, \mathsf{NR}^*, \Phi_{\mathsf{aff}}, \vec{\omega})$ has at most one 1 bit and every output of H has at least two 1 bits. Note that in particular the output of H is never 0^n, so it is always in the domain of NR^*.

Lemma 4.2. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let NR^* be defined via $\mathsf{NR}^*(\vec{a}, x) = [\prod_{i=1}^n a_i^{x_i}]$, where $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. Let $\mathscr{D}$ be an adversary against the Φ_{aff}-key-collision security of NR^* that makes $Q_{\mathscr{D}}$ oracle queries. Then we can construct an adversary $\mathscr{C}$ against the* DL *problem in $\mathbb{G}$ with the same running time as that of $\mathscr{D}$ such that*

$$\mathbf{Adv}^{\mathsf{kc}}_{\Phi_{\mathsf{aff}}, \mathsf{NR}^*}(\mathscr{D}) \;\leq\; n \cdot \mathbf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathscr{C}).$$

Since the hardness of DDH implies the hardness of DL, the above lemma does not introduce any additional hardness assumptions beyond DDH.

Proof of Lemma 4.2. Let $\mathscr{D}$ be an adversary against the Φ_{aff}-key-collision security of NR^* that makes $Q_{\mathscr{D}}$ oracle queries. Then we construct an adversary $\mathscr{C}$ against the DL problem in $\mathbb{G}$ as follows. Adversary $\mathscr{C}$ receives as input a DL tuple $([1], [a])$. Adversary $\mathscr{C}$ then picks $j \xleftarrow{\$} \{1, \ldots, n\}$ at random; this is a guess of a coordinate where the two vectors of affine functions $\vec{\phi}^{(1)}$ and $\vec{\phi}^{(2)}$ that $\mathscr{D}$ will use as inputs in the **Finalize** procedure are different. Then $\mathscr{C}$ picks $a_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1, \ldots, n, i \neq j$ at random. Adversary $\mathscr{C}$ implicitly sets $a_j = a$.

When $\mathscr{D}$ makes a query (ϕ, x), $\mathscr{C}$ computes $y = ([\phi_j(a_j)^{x_j}])^{\prod_{\substack{i=1 \\ i \neq j}}^n \phi_i(a_i)^{x_i}} = [\prod_{i=1}^n \phi_i(a_i)^{x_i}] = \mathsf{NR}^*(\vec{a}, x)$, where $\vec{a} = (a_1, \ldots, a_n)$. Here, $\mathscr{C}$ uses its input $[a]$ to compute an "affine function in the exponent" for $[\phi_j(a_j)]$. At the end, $\mathscr{D}$ sends $(\vec{\phi}^{(1)}, \vec{\phi}^{(2)})$ to $\mathscr{C}$ and $\mathscr{D}$ wins if $\vec{\phi}^{(1)} \neq \vec{\phi}^{(2)}$ and $\vec{\phi}^{(1)}(\vec{a}) = \vec{\phi}^{(2)}(\vec{a})$, where $\vec{\phi}^{(i)} = (\phi_1^{(i)}, \ldots, \phi_n^{(i)}), i \in \{1,2\}$. Since j was chosen uniformly at random and $\vec{\phi}^{(1)} \neq \vec{\phi}^{(2)}$, with probability at least $\frac{1}{n}$, we have $\phi_j^{(1)} \neq \phi_j^{(2)}$ but $\phi_j^{(1)}(a_j) = \phi_j^{(2)}(a_j)$. In this case, $a_j = a$ is the root of the non zero affine function $\psi = \phi_j^{(1)} - \phi_j^{(2)}$, that can be easily computed. Hence, we have

$$\mathbf{Adv}^{\mathsf{kc}}_{\Phi_{\mathsf{aff}}, \mathsf{NR}^*}(\mathscr{D}) \;\leq\; n \cdot \mathbf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathscr{C})$$

and the claim follows. □

proc Initialize // G_i, $i = 0, \ldots, n$	**proc RKFn**(ϕ, x) // G_i, $i = 0, \ldots, n$
$f \xleftarrow{\$} \mathsf{Fun}(\{0,1\}^n \setminus \{0^n\}, \mathbb{G})$	If $\mathsf{hw}(x) \leq n - i$ then
proc Finalize(b')	$\quad y \leftarrow \mathsf{KT}^f_{\Phi_{\mathsf{aff}}}(\phi, x)$
Return $b' = b$	Else $y \xleftarrow{\$} \mathbb{G}$
	Return y

Figure 4: Games for the proof of Lemma 4.3.

Lemma 4.3. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let NR^* be defined via $\mathsf{NR}^*(\vec{a}, x) = [\prod_{i=1}^n a_i^{x_i}]$, where $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. Let $\mathsf{KT}_{\Phi_{\mathsf{aff}}}$ be defined via*

$$\mathsf{KT}^f_{\Phi_{\mathsf{aff}}}(\phi, x) = \left[\prod_{i \in S(x)} c_i\right] \cdot \prod_{y \preceq x, y \neq 0^n} f(y)^{\prod_{j \in S(y)} b_j \prod_{k \in S(x) \setminus S(y)} c_k}$$

where $\vec{\phi} = (\phi_1, \ldots, \phi_n) \in \Phi_{\mathsf{aff}}$, with ϕ_i: $a \in \mathbb{Z}_p \mapsto b_i a + c_i \in \mathbb{Z}_p$, $b_i \neq 0$, for $i = 1, \ldots, n$. Then $\mathsf{KT}_{\Phi_{\mathsf{aff}}}$ is a key-transformer for $(\mathsf{NR}^*, \Phi_{\mathsf{aff}})$. Moreover, the worst-case running time of this key-transformer is the time required to compute $O(2^n)$ exponentiations in $\mathbb{G}$.

Proof of Lemma 4.3. Let us first check the correctness condition.

$$\begin{aligned}
\mathsf{KT}^{\mathsf{NR}^*(\vec{a},\cdot)}_{\Phi_{\mathsf{aff}}}(\vec{\phi}, x) &= \left[\prod_{i \in S(x)} c_i\right] \cdot \prod_{y \preceq x, y \neq 0^n} \left[\prod_{l \in S(y)} a_l\right]^{\prod_{j \in S(y)} b_j \prod_{k \in S(x) \setminus S(y)} c_k} \\
&= \prod_{R \subseteq S(x)} \left[\prod_{i \in R} (b_i \cdot a_i) \prod_{j \in S(x) \setminus R} c_j\right] = \left[\sum_{R \subseteq S(x)} \prod_{i \in R} (b_i \cdot a_i) \prod_{j \in S(x) \setminus R} c_j\right] \\
&= \left[\prod_{i \in S(x)} (b_i \cdot a_i + c_i)\right] = \left[\prod_{i=1}^n (b_i \cdot a_i + c_i)^{x_i}\right] = \mathsf{NR}^*(\vec{\phi}(\vec{a}), x) \ .
\end{aligned}$$

Then, we have verified the correctness condition, and it is clear that the worst-case running time is the time to compute 2^n exponentiations in $\mathbb{G}$, when $x = 11 \| \ldots \| 1$ and none of the exponents is 0. Hence, only the uniformity condition remains to prove. We use the sequence of games in Figure 4. Let us recall that the adversary is supposed to be unique-input, meaning that for any sequence of queries $(\vec{\phi}_1, x_1), \ldots, (\vec{\phi}_q, x_q)$, the entries x_i, for $i = 1, \ldots, q$ are all distinct. We denote by hw(x) the Hamming weight of a bitstring x. Let $\textsc{Succ}_i$ denote the event that game G_i output takes the value 1.

In game G_0, the "If" statement will always pass since $\mathsf{hw}(x) \leq n$ for any bitstring of length n. Hence, we have

$$\Pr[\textsc{Succ}_0] = \Pr\left[\mathrm{KTReal}^{\mathscr{A}}_{\mathsf{KT}} \Rightarrow 1\right] .$$

We claim that for all $0 \leq i \leq n - 1$,

$$\Pr[\textsc{Succ}_i] = \Pr[\textsc{Succ}_{i+1}] .$$

The only difference between games G_i and G_{i+1} is in the way that bitstrings x of Hamming weight $n - i$ are handled. Indeed, such a string is fed to $\mathsf{KT}^f_{\Phi_{\mathsf{aff}}}(\vec{\phi}, x)$ in G_i, which computes

$$\mathsf{KT}^f_{\Phi_{\mathsf{aff}}}(\vec{\phi}, x) = \left[\prod_{i \in S(x)} c_i\right] \cdot \prod_{y \preceq x, y \neq 0^n} f(y)^{\prod_{j \in S(y)} b_j \prod_{k \in S(x) \setminus S(y)} c_k}$$

where $\vec{\phi} = (\phi_1, \ldots, \phi_n) \in \Phi_{\mathsf{aff}}$, with ϕ_i: $\vec{T} \mapsto b_i T_i + c_i$, $b_i \neq 0$, for $i = 1, \ldots, n$. Now, since we need only deal with unique-input adversaries, this is the only time that G_i will query f at input x (all other queries to f will be at other points with the same Hamming weight or at points with strictly smaller Hamming weight). Hence, the entire value computed above can equivalently be set to a value chosen uniformly at random. (This relies on the exponent for $f(x)$ used in the computation being non-zero; this is guaranteed by the requirement that $b_i \neq 0$, for $i = 1, \ldots, n$ and the fact that when $y = x$, the product $\prod_{k \in S(x) \setminus S(y)} c_k$ is empty.) Setting the entire value to a uniformly random value is exactly what is done in G_{i+1}, and the claim follows.

Finally, in G_n, the "If" statement will never pass since $\mathsf{hw}(x) > 0$ for any $x \in \{0,1\}^n \setminus \{0\}^n$, so we have

$$\Pr\left[\,\textsc{Succ}_n\,\right] = \Pr\left[\,\mathrm{KTRand}^{\mathscr{A}}_{\mathsf{KT}} \Rightarrow 1\,\right].$$

The uniformity condition follows. □

Lemma 4.4. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Let $\mathscr{A}$ be an adversary against the Φ_{aff}-statistical-key-collision security for $\mathsf{Fun}(\mathbb{Z}_p^n, \{0,1\}^n, \mathbb{G})$ making $Q_{\mathscr{A}}$ queries. Then we have*

$$\mathbf{Adv}^{\mathsf{skc}}_{\Phi_{\mathsf{aff}}}(\mathscr{A}) \;\leq\; \frac{Q_{\mathscr{A}}^2}{2p}.$$

Proof of Lemma 4.4. Let $\mathscr{A}$ be an adversary against the Φ_{aff}-statistical-key-collision security for $\mathsf{Fun}(\mathbb{Z}_p^n, \{0,1\}^n, \mathbb{G})$ that makes $Q_{\mathscr{A}}$ queries. Since the function F defined in the **Initialize** procedure is a random function, $\mathscr{A}$ does not learn any information on the key $\vec{a}$ until $b' \leftarrow 1$, so $\mathbf{Adv}^{\mathsf{skc}}_{\Phi_{\mathsf{aff}}}(\mathscr{A})$ is bounded by the probability that $\mathscr{A}$ makes use in its queries of two different RKD functions that lead to the same key. We claim that

$$\mathbf{Adv}^{\mathsf{skc}}_{\Phi_{\mathsf{aff}}}(\mathscr{A}) \;\leq\; \frac{Q_A^2}{2p}.$$

This follows easily on noting that, if two different RKD functions lead to the same key, then those two functions must differ in some coordinate k. This means that the difference in those components is a non-constant affine function ψ_k such that $\psi_k(a_k) = 0$, where a_k is the k-th component of key $\vec{a}$ that was taken uniformly at random in the **Initialize** procedure. Since ψ_k is a non-constant affine function and a_k is uniformly random in $\mathbb{Z}_p$, the probability that $\psi_k(a_k) = 0$ is bounded by $\frac{1}{p}$. To obtain the final result, one simply applies a union bound over the (at most) $\binom{Q_{\mathscr{A}}}{2}$ pairs of choices of different RKD functions accessed by $\mathscr{A}$. □

We now have everything we need to apply Theorem 3.1 to NR^*. Combining Theorem 3.1, Lemmas 4.1–4.4 and the above properties, we obtain the following theorem.

Theorem 4.5. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let NR^* be defined via $\mathsf{NR}^*(\vec{a}, x) = [\prod_{i=1}^n a_i^{x_i}]$, where $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. Let $\overline{\mathcal{D}} = \{0,1\}^n \times \mathbb{G}^n$ and let h: $\overline{\mathcal{D}} \to \{0,1\}^{n-2}$ be a hash function. Let $\omega_i = 0^{i-1} \,\|\, 1 \,\|\, 0^{n-i}$, for $i = 1, \ldots, n$. Define F: $\mathbb{Z}_p^n \times \{0,1\}^n \to \mathbb{G}$ by*

$$F(\vec{a}, x) = \mathsf{NR}^*(\vec{a}, 11 \,\|\, h(x, \mathsf{NR}^*(\vec{a}, \vec{\omega})))$$

for all $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n$. Let $\mathscr{A}$ be a Φ_{aff}-restricted adversary against the prf-rka security of F that makes $Q_{\mathscr{A}}$ oracle queries. Then we can construct an adversary $\mathscr{B}$ against the DDH *problem in $\mathbb{G}$, an adversary $\mathscr{C}$ against the cr security of h, and an adversary $\mathscr{D}$ against the* DL *problem in $\mathbb{G}$, such that*

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_{\mathsf{aff}},F}(\mathscr{A}) \;\leq\; n \cdot \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathscr{B}) + \mathbf{Adv}^{\mathsf{cr}}_{h}(\mathscr{C}) + n \cdot \mathbf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathscr{D}) + \frac{Q_{\mathscr{A}}^2}{2p}.$$

The running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time required to compute $O(Q_{\mathscr{A}} \cdot (n+1) \cdot 2^n)$ exponentiations in $\mathbb{G}$. The running times of $\mathscr{C}$ and $\mathscr{D}$ are the same as that of $\mathscr{A}$.

5 Further Generalisation of the Bellare-Cash Framework

We introduce a new type of PRF, called an $(\mathcal{S}, \Phi)$-Unique-Input-RKA-PRF. We then use this notion as a tool in a further extension of the Bellare-Cash framework that can be applied to *non*-key-malleable PRFs and *non*-claw-free classes of RKD functions. This new framework provides in particular a route to proving that the variant of the Naor-Reingold PRF introduced in Section 4 is actually Φ_d-RKA-secure.

$(\mathcal{S}, \Phi)$-Unique-Input-RKA-PRF. Let $M\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions. Let $\mathcal{S}$ be a subset of $\mathcal{D}$ and Φ be a class of RKD functions. We consider the class of adversaries $\mathscr{A}$ in Figure 5 such that all queries (ϕ, x) with $x \in \mathcal{S}$ made by $\mathscr{A}$ to its oracle are for distinct values of x. That is, for any sequence of $\mathscr{A}$'s queries $(\phi_1, x_1), \ldots, (\phi_q, x_q)$ with $x_i \in \mathcal{S}$ for all $i = 1, \ldots, q$, we require all the x_i to be distinct (no such restriction is made for queries (ϕ_i, x_i) with $x_i \notin \mathcal{S}$). We denote the advantage of such an adversary $\mathscr{A}$ by $\mathbf{Adv}_M^{(\mathcal{S},\Phi)\text{-ui-prf-rka}}(\mathscr{B})$. We then say that M is an $(\mathcal{S}, \Phi)$-unique-input-RKA-PRF if the advantage of any such Φ-restricted, efficient adversary $\mathscr{A}$ in attacking $(\mathcal{S}, \Phi)$-unique-input-prf-rka security is negligible.

proc Initialize
$K \xleftarrow{\$} \mathcal{K}$; $b \xleftarrow{\$} \{0,1\}$
proc Finalize(b')
Return $b' = b$
	proc RKFn(ϕ, x)
	If $x \in \mathcal{S}$ then
	  If $b = 0$ then $y \leftarrow M(\phi(K), x)$
	  Else $y \xleftarrow{\$} \mathcal{R}$
	Else $y \leftarrow M(\phi(K), x)$
	Return y

Figure 5: Game defining the $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of a PRF M.

The following theorem is an analogue of Theorem 3.1 in which the roles of key malleability and hash function compatibility are replaced by our new notion, $(\mathcal{S}, \Phi)$-unique-input-prf-rka security.

Theorem 5.1. *Let $M\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions and Φ be a class of* RKD *functions. Let $\vec{\omega} \in \mathcal{D}^m$ be a strong key fingerprint for M. Let $\overline{\mathcal{D}} = \mathcal{D} \times \mathcal{R}^m$ and let $H\colon \overline{\mathcal{D}} \to \mathcal{S}$ be a hash function, where $\mathcal{S} \subseteq \mathcal{D} \setminus \{\omega_1, \ldots, \omega_m\}$. Define $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ by*

$$F(K, x) = M(K, H(x, M(K, \vec{\omega})))$$

for all $K \in \mathcal{K}$ and $x \in \mathcal{D}$. Let $\mathscr{A}$ be a Φ-restricted adversary against the prf-rka security of F that makes $Q_{\mathscr{A}} \leq |\mathcal{S}|$ oracle queries. Then we can construct an adversary $\mathscr{B}$ against the $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of M, an adversary $\mathscr{C}$ against the cr security of H, an adversary $\mathscr{D}$ against the Φ-kc security of M and an adversary $\mathscr{E}$ against Φ-skc security for $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ such that

$$\mathbf{Adv}_{\Phi,F}^{\text{prf-rka}}(\mathscr{A}) \;\leq\; \mathbf{Adv}_M^{(\mathcal{S},\Phi)\text{-ui-prf-rka}}(\mathscr{B}) + \mathbf{Adv}_H^{\text{cr}}(\mathscr{C}) + \mathbf{Adv}_{\Phi,M}^{\text{kc}}(\mathscr{D}) + \mathbf{Adv}_{\Phi}^{\text{skc}}(\mathscr{E}). \qquad (2)$$

Adversaries $\mathscr{C}$, $\mathscr{D}$ and $\mathscr{E}$ have the same running time as $\mathscr{A}$. Adversary $\mathscr{B}$ makes $(m+1) \cdot Q_A$ oracle queries and has the same running time as $\mathscr{A}$.

Overview of the Proof. The proof of the above theorem is detailed below and relies on the sequence of 10 games (games $\mathrm{G}_0 - \mathrm{G}_9$) described in Figure 6. Here we provide a brief overview. Since the RKD functions that we consider in our case may have claws, we start by dealing with possible collisions on the related-keys in the RKPRFReal case, using the key-collision notion(games $\mathrm{G}_0 - \mathrm{G}_2$). Then, in games $\mathrm{G}_3 - \mathrm{G}_4$, we deal with possible collisions on hash values in order to ensure that the hash values h used to compute the output y are distinct. Then, in contrast to the proof of Theorem 3.1, we use the new $(\mathcal{S}, \Phi)$-unique-input-RKA-PRF notion and the compatibility condition to show that it is

hard to distinguish the output of F from a uniformly random output(games $G_5 - G_6$). Finally, we use the statistical-key-collision security notion to deal with possible key collisions in the RKPRFRand case (games $G_7 - G_9$) so that G_9 matches the description of the RKPRFRand Game.

Remark 5.2. In Appendix A, we explore the relationship between key-malleable PRFs and unique-input-RKA-secure PRFs. Specifically, we show that the $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of a Φ-key-malleable PRF M is implied by its regular prf security if the key-transformer KT associated with M satisfies a new condition that we call $\mathcal{S}$-uniformity. This condition demands that the usual uniformity condition for KT should hold on the subset $\mathcal{S}$ of $\mathcal{D}$ rather than on all of $\mathcal{D}$. Whether $\mathcal{S}$-uniformity is implied by (regular) uniformity is an open question.

Proof of Theorem 5.1. The proof is based on the sequence of games in Figure 6. Much of the proof is similar to the proof of Theorem 3.1 (which itself is based on the proof of the general framework of Bellare and Cash from [BC10a]). The current proof, however, is somewhat simpler and has fewer games since it relies on a stronger security property of the underlying PRF M, namely its $(\mathcal{S}, \Phi)$-unique-input-prf-rka security. Let $\textsc{Succ}_i$ denote the event that game G_i output takes the value 1.

Game G_1 introduces storage of used RKD-functions and values of $\vec{\omega}$ in sets D and E respectively and sets flag_1 to true if the same value of $\vec{\omega}$ arises for two different RKD-functions. Since this storage does not affect the values returned by **RKFn**

$$\Pr[\,\textsc{Succ}_1\,] = \Pr[\,\textsc{Succ}_0\,] .$$

Game G_2 adds the boxed code which changes how the repetition of an $\vec{\omega}$ value is handled, by picking instead a random value from $\mathcal{R}^m \setminus E$ that will not repeat any previous one. Games G_1 and G_2 are identical until flag_1 is set to true, hence we have

$$\Pr[\,\textsc{Succ}_1\,] \;\leq\; \Pr[\,\textsc{Succ}_2\,] + \Pr[\,E_1\,]$$

where E_1 denotes the event that the execution of $\mathscr{A}$ with game G_1 sets flag_1 to true. We design an adversary $\mathscr{D}$ attacking the Φ-key-collision security of M such that

$$\Pr[\,E_1\,] \;\leq\; \mathbf{Adv}^{\mathsf{kc}}_{\Phi,M}(\mathscr{D}) .$$

Adversary $\mathscr{D}$ runs $\mathscr{A}$. When the latter makes a **RKFn**-query (ϕ, x), adversary $\mathscr{D}$ queries (ϕ, ω_i), for $i = 1, \ldots, |\vec{\omega}|$, to its oracle, then computes $\vec{\omega}$ and then $h = H(x, \vec{\omega})$ and finally queries (ϕ, h) to its oracle and sends it to $\mathscr{A}$. When $\mathscr{A}$ stops, $\mathscr{D}$ searches for two different RKD-functions ϕ queried by $\mathscr{A}$ that lead to the same value $\vec{\omega}$ and returns these two functions if found. Since $\vec{\omega}$ is a strong key fingerprint, two such functions lead to the same key, so $\mathscr{D}$ wins if he finds such two functions. (Of course, if the class of RKD-functions is claw-free, the advantage of the attacker is 0.)

Game G_3 introduces the storage of hash values in a set G and sets flag_2 to true if the same hash output arises twice. Since this storage does not affect the values returned by **RKFn**, we have

$$\Pr[\,\textsc{Succ}_3\,] = \Pr[\,\textsc{Succ}_2\,] .$$

Game G_4 adds the boxed code which changes how repetition of hash values is handled, by picking instead a random value h from $\mathcal{S} \setminus G$ that will not repeat any previously used hash value. Games G_3 and G_4 are identical until flag_2 is set to true, hence we have

$$\Pr[\,\textsc{Succ}_3\,] \;\leq\; \Pr[\,\textsc{Succ}_4\,] + \Pr[\,E_2\,]$$

proc Initialize // G_0
$K \xleftarrow{\$} \mathcal{K}$

proc RKFn(ϕ, x) // G_0
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\varpi_i \leftarrow M(\phi(K), \omega_i)$
$h \leftarrow H(x, \vec{\varpi})$
$y \leftarrow M(\phi(K), h)$
Return y

proc Finalize(b') // All Games
Return b'

---

proc Initialize // $\mathrm{G}_1, \mathrm{G}_2$
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$

proc RKFn(ϕ, x) // G_1, $\boxed{\mathrm{G}_2}$
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\varpi_i \leftarrow M(\phi(K), \omega_i)$
If $\vec{\varpi} \in E$ and $\phi \notin D$ then
  $\mathsf{flag}_1 \leftarrow \mathsf{true}$; $\boxed{\vec{\varpi} \xleftarrow{\$} \mathcal{R}^m \setminus E}$
$\boxed{\text{Else}}$ $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\varpi}\}$
$h \leftarrow H(x, \vec{\varpi})$
$y \leftarrow M(\phi(K), h)$
Return y

---

proc Initialize // $\mathrm{G}_3, \mathrm{G}_4$
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$; $G \leftarrow \emptyset$

proc RKFn(ϕ, x) // G_3, $\boxed{\mathrm{G}_4}$
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\varpi_i \leftarrow M(\phi(K), \omega_i)$
If $\vec{\varpi} \in E$ and $\phi \notin D$ then
  $\vec{\varpi} \xleftarrow{\$} \mathcal{R}^m \setminus E$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\varpi}\}$
$h \leftarrow H(x, \vec{\varpi})$
If $h \in G$ then $\mathsf{flag}_2 \leftarrow \mathsf{true}$
  $\boxed{h \xleftarrow{\$} \mathcal{S} \setminus G}$
$G \leftarrow G \cup \{r\}$
$y \leftarrow M(\phi(K), h)$
Return y

---

proc Initialize // G_5
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$; $G \leftarrow \emptyset$

proc RKFn(ϕ, x) // G_5
For $i = 1, \ldots, |\vec{\omega}|$ do
  $\varpi_i \leftarrow M(\phi(K), \omega_i)$
If $\vec{\varpi} \in E$ and $\phi \notin D$ then
  $\vec{\varpi} \xleftarrow{\$} \mathcal{R}^m \setminus E$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\varpi}\}$
$h \leftarrow H(x, \vec{\varpi})$
If $h \in G$ then $h \xleftarrow{\$} \mathcal{S} \setminus G$
$G \leftarrow G \cup \{r\}$
$y \xleftarrow{\$} \mathcal{R}$
Return y

---

proc Initialize // G_6
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$
$G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$

proc RKFn(ϕ, x) // G_6
$y \xleftarrow{\$} \mathcal{R}$
Return y

---

proc Initialize // G_9
$K \xleftarrow{\$} \mathcal{K}$; $G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$

proc RKFn(ϕ, x) // G_9
$y \leftarrow G(\phi(K), x)$
Return y

---

proc Initialize // $\boxed{\mathrm{G}_7}$, G_8
$K \xleftarrow{\$} \mathcal{K}$; $D \leftarrow \emptyset$; $E \leftarrow \emptyset$
$G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$

proc RKFn(ϕ, x) // $\boxed{\mathrm{G}_7}$, G_8
$y \leftarrow G(\phi(K), x)$
If $\phi(K) \in E$ and $\phi \notin D$ then
  $\mathsf{flag}_3 \leftarrow \mathsf{true}$; $\boxed{y \xleftarrow{\$} \mathcal{R}}$
$D \leftarrow D \cup \{\phi\}$; $E \leftarrow E \cup \{\phi(K)\}$
Return y

Figure 6: Games for the proof of Theorem 5.1.

where E_2 denotes the event that the execution of $\mathscr{A}$ with game G_3 sets flag_2 to true. We design an adversary $\mathscr{C}$ attacking the cr-security of H such that

$$\Pr[E_2] \leq \mathbf{Adv}^{\mathsf{cr}}_H(\mathscr{C}) .$$

Adversary $\mathscr{C}$ starts by picking $K \xleftarrow{\$} \mathcal{K}$ and initializes $j \leftarrow 0$. It runs $\mathscr{A}$. When the latter makes a **RKFn**-query (ϕ, x), adversary $\mathscr{C}$ responds via

For $i = 0, \ldots, |\vec{\omega}|$ do: $\overline{\omega}_i \leftarrow M(\phi(K), \omega_i)$
$j \leftarrow j + 1$; $\phi_j \leftarrow \phi$; $x_j \leftarrow x$
If $\vec{\omega} \in E$ and $\phi \notin D$ then $\vec{\omega} \xleftarrow{\$} \mathcal{S} \setminus E$ $\qquad (*)$
Else $D \leftarrow D \cup \{\phi\}$
$E \leftarrow E \cup \{\vec{\omega}\}$
$w_j \leftarrow \vec{\omega}$
$h \leftarrow H(x, \vec{\omega})$
$h_j \leftarrow h$
$y \leftarrow M(\phi(K), h)$
Return y.

When $\mathscr{A}$ halts, $\mathscr{C}$ searches for a, b satisfying $1 \leq a < b \leq j$ such that $h_a = h_b$ and, if it finds them, outputs $(x_a, w_a), (x_b, w_b)$ and halts. The pairs (x_a, w_a) and (x_b, w_b) are distinct. Indeed, consider two cases: first, if $\phi_a = \phi_b$ then since $\mathscr{A}$ never repeats an oracle query, $x_a \neq x_b$ hence $(x_a, w_a) \neq (x_b, w_b)$. Second, if $\phi_a \neq \phi_b$, then condition $(*)$ ensures that $w_a \neq w_b$. Hence once again, $(x_a, w_a) \neq (x_b, w_b)$, and then

$$\Pr[\textsc{Succ}_3] \leq \Pr[\textsc{Succ}_4] + \mathbf{Adv}^{\mathsf{cr}}_H(\mathscr{C}) .$$

In game G_5, instead of returning the value $M(\phi(K), h)$, we always return a random value. To show that games G_4 and G_5 are indistinguishable, we design an adversary $\mathscr{B}$ against the $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of M such that

$$\Pr[\textsc{Succ}_4] \leq \Pr[\textsc{Succ}_5] + \mathbf{Adv}^{(\mathcal{S},\Phi)\text{-ui-prf-rka}}_M(\mathscr{B}) .$$

Adversary $\mathscr{B}$ starts by initializing sets $D \leftarrow \emptyset$, $E \leftarrow \emptyset$, $G \leftarrow \emptyset$. Then $\mathscr{B}$ runs $\mathscr{A}$. When the latter makes an **RKFn**-query (ϕ, x), $\mathscr{B}$ responds as follows. For $i = 1, \ldots, |\vec{\omega}|$, it asks (ϕ, ω_i) to its oracle and sets $\overline{\omega}_i$ to this value. Since for all $i = 1, \ldots, |\vec{\omega}|$, $\omega_i \notin \mathcal{S}$ by assumption, the value it gets is $M(\phi(K), \omega_i)$, whatever its oracle is. Then, $\mathscr{B}$ checks if $\vec{\omega} \in E$ and $\phi \notin D$. If they do, $\mathscr{B}$ picks $\vec{\omega} \xleftarrow{\$} \mathcal{R}^m \setminus E$ at random, otherwise $\mathscr{B}$ sets $D \leftarrow D \cup \{\phi\}$. $\mathscr{B}$ then sets $E \leftarrow E \cup \{\vec{\omega}\}$. Next, $\mathscr{B}$ computes $h \leftarrow H(x, \vec{\omega})$ and checks if $h \in G$. If it does, $\mathscr{B}$ picks $h \xleftarrow{\$} \mathcal{S} \setminus G$ at random. Notice that this step guarantees that all values h are in $\mathcal{S}$ and are all distinct as long as $\mathscr{A}$ makes at most $|\mathcal{S}|$ queries. Finally, $\mathscr{B}$ sets $G \leftarrow G \cup \{h\}$, makes the query (ϕ, h) to its oracle, and returns the value it gets, which is either $M(\phi(K), h)$ or a uniformly random value, to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. The definition of $\mathcal{S}$ ensures that it does not contain any ω_i for $i = 1, \ldots, |\vec{\omega}|$. It follows from these observations that $\mathscr{B}$ is a unique-input adversary for queries in $\mathcal{S}$. Finally, it is clear that if $\mathscr{B}$'s oracle gives real outputs of M for queries in $\mathcal{S}$, then it simulates exactly game G_4 and if $\mathscr{B}$'s oracle gives uniformly random values for queries in $\mathcal{S}$, then it simulates exactly game G_5.

In game G_6, we simply set the value y to a uniformly random value. Clearly, G_5 and G_6 are identical since the value returned is a uniformly random value for any query. Then, we have

$$\Pr[\textsc{Succ}_5] = \Pr[\textsc{Succ}_6] .$$

In game G_7, we check if two different queries can lead to a key collision. Since the "If" test ensures that the returned value is still uniformly random over $\mathcal{R}$ even when two different queries result in the same key, games G_6 and G_7 are identical. Hence,

$$\Pr[\,\text{SUCC}_6\,] = \Pr[\,\text{SUCC}_7\,] .$$

In game G_8, we compute the output of **RKFn** using a random function G in $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$. Since games G_7 and G_8 are identical until flag_3 is set to true, we have

$$\Pr[\,\text{SUCC}_7\,] \;\le\; \Pr[\,\text{SUCC}_8\,] + \Pr[\,E_3\,]$$

where E_3 denotes the event that the execution of $\mathscr{A}$ with game G_8 sets flag_3 to true. To bound the probability of event E_3, we design an adversary $\mathscr{E}$ attacking Φ-statistical-key-collision security for $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ such that

$$\Pr[\,E_3\,] \;\le\; \mathbf{Adv}^{\mathsf{skc}}_{\Phi}(\mathscr{E}) .$$

Adversary $\mathscr{E}$ runs $\mathscr{A}$. When the latter makes an **RKFn**-query (ϕ, x), so does $\mathscr{E}$ and $\mathscr{E}$ returns the value he receives to $\mathscr{A}$. When $\mathscr{A}$ stops, if $\mathscr{A}$ has queried two different functions ϕ_1 and ϕ_2 such that $\phi_1(K) = \phi_2(K)$ then b' was set to 1 when the second of these two functions was queried by $\mathscr{E}$, and then $\mathscr{E}$ wins. (Of course, if the class of RKD functions is claw-free, this probability is 0.)

Since $\mathscr{A}$ does not repeat oracle queries and since key collisions are dealt with in a similar way, it follows that games G_8 and G_9 are identical. Thus,

$$\Pr[\,\text{SUCC}_8\,] = \Pr[\,\text{SUCC}_9\,] .$$

Equation (2) on page 17 now follows by combining the bounds arising in the different game hops. □

6 Related-Key Security for Polynomial RKD Functions

We apply Theorem 5.1 to the variant NR^* of the Naor-Reingold PRF for the class of RKD functions $\Phi_d = \{\phi\colon\ \mathcal{K} \to \mathcal{K} \mid \phi_i\colon\ \vec{T} \mapsto \sum_{j=0}^{d} \alpha_{i,j} \cdot T^j, (\alpha_{i,1}, \ldots, \alpha_{i,d}) \neq 0^d; \forall i = 1, \ldots, n\}$. Specifically, we prove that NR^* can be used to build a Φ_d-RKA-secure PRF, under the d-DDHI assumption. Remarkably, our proof provides an efficient reduction, avoiding an exponential running time like that seen in Theorem 4.5. The key step in establishing our result is Lemma 6.3. Its proof involves at its core the construction of a bespoke key-transformer to handle Φ_d and a delicate analysis of it using sequences of hybrid games.

In what follows, we prove the various properties needed to apply Theorem 5.1 to NR^*.

Strong Key Fingerprint. Let $\omega_i = 0^{i-1} \,\|\, 1 \,\|\, 0^{n-i}$, for $i = 1, \ldots, n$. Then, as before, $\vec{\omega}$ is a strong key fingerprint for NR^*.

Hash Function. Let $\overline{\mathcal{D}} = \{0,1\}^n \times \mathbb{G}^n$ and let $h\colon\ \overline{\mathcal{D}} \to \{0,1\}^{n-2}$ be a collision resistant hash function. Then, as previously, the hash function defined by $H(x, \vec{z}) = 11 \,\|\, h(x, \vec{z})$ is a collision resistant hash function with range $\mathcal{S}$ satisfying $\mathcal{S} \subseteq \{0,1\}^n \setminus (\{\omega_1, \ldots, \omega_n\} \cup \{0^n\})$.

Lemma 6.1. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let NR^* be defined via $\mathsf{NR}^*(\vec{a}, x) = [\prod_{i=1}^{n} a_i^{x_i}]$, where $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. Let $\mathscr{D}$ be an adversary against the Φ_d-key-collision security of NR^* that makes $Q_{\mathscr{D}}$ oracle queries. Then we can construct an adversary $\mathscr{C}$ against the d-SDL problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}^{\mathsf{kc}}_{\Phi_d, \mathsf{NR}^*}(\mathscr{D}) \le n \cdot \mathbf{Adv}^{d\text{-}\mathsf{sdl}}_{\mathbb{G}}(\mathscr{C}).$$

The running time of $\mathscr{C}$ is that of $\mathscr{D}$ plus the time required to factorize a polynomial of degree at most d in $\mathbb{F}_p$ (sub-quadratic in d and logarithmic in p) plus $O(Q_{\mathscr{D}} \cdot d)$ exponentiations in $\mathbb{G}$.

Proof of Lemma 6.1. Let $\mathscr{D}$ be an adversary against the Φ_d-key-collision security of NR^* that makes $Q_{\mathscr{D}}$ oracle queries. Then we construct an adversary $\mathscr{C}$ against the d-SDL problem in $\mathbb{G}$ as follows.

Adversary $\mathscr{C}$ receives as input a d-SDL tuple $([1], [a], \ldots, [a^d])$ where $a \xleftarrow{\$} \mathbb{Z}_p$. Adversary $\mathscr{C}$ then picks $j \xleftarrow{\$} \{1, \ldots, n\}$ at random; this is a guess of a coordinate where the two vectors of polynomial functions $\vec{\phi}^{(1)}$ and $\vec{\phi}^{(2)}$ that $\mathscr{D}$ will use as inputs in the **Finalize** procedure are different. Then $\mathscr{C}$ picks $a_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1, \ldots, n, i \neq j$ at random. Adversary $\mathscr{C}$ implicitly set $a_j = a$.

When $\mathscr{D}$ makes a query $(p\vec{h}i, x)$, $\mathscr{C}$ computes $y = ([\phi_j(a_j)^{x_j}])^{\prod_{\substack{i=1 \\ i \neq j}}^{n} \phi_i(a_i)^{x_i}} = [\prod_{i=1}^{n} \phi_i(a_i)^{x_i}] = \mathsf{NR}^*(\vec{a}, x)$, where $\vec{a} = (a_1, \ldots, a_n)$. Here, $\mathscr{C}$ uses its input $([1], [a], \ldots, [a^d])$ to compute a "polynomial function in the exponent" for $[\phi_j(a_j)]$.

At the end, $\mathscr{D}$ sends $(\vec{\phi}^{(1)}, \vec{\phi}^{(2)})$ to $\mathscr{C}$ and $\mathscr{D}$ wins if $\vec{\phi}^{(1)} \neq \vec{\phi}^{(2)}$ and $\vec{\phi}^{(1)}(\vec{a}) = \vec{\phi}^{(2)}(\vec{a})$, where $\vec{\phi}^{(i)} = (\phi_1^{(i)}, \ldots, \phi_n^{(i)}), i \in \{1, 2\}$. Since j was chosen uniformly at random and $\vec{\phi}^{(1)} \neq \vec{\phi}^{(2)}$, with probability at least $\frac{1}{n}$, we have $\phi_j^{(1)} \neq \phi_j^{(2)}$ but $\phi_j^{(1)}(a_j) = \phi_j^{(2)}(a_j)$. In this case $a_j = a$ is a root of the polynomial $\psi := \phi_j^{(1)} - \phi_j^{(2)}$, whose degree is at most d. So $\mathscr{D}$ factorizes ψ (using, for instance, the Kedlaya-Umans algorithm [KU11], which has complexity sub-quadratic in d and logarithmic in p), and selects as its output the unique root r such that $[r] = [a]$. The claim follows. □

Lemma 6.2. *Let $\mathbb{G}$ be a group of prime order p. Let $\mathsf{Fun}(\mathbb{Z}_p^n, \{0,1\}^n \setminus \{0^n\}, \mathbb{G})$ be the set of functions from which the random function in the Φ_d-statistical-key-collision security game is taken. Let $\mathscr{A}$ be an adversary against the Φ_d-statistical-key-collision security that makes $Q_{\mathscr{A}}$ queries. Then we have*

$$\mathbf{Adv}^{\mathsf{skc}}_{\Phi_d}(\mathscr{A}) \leq \frac{d \cdot Q_{\mathscr{A}}^2}{2p}.$$

Proof of Lemma 6.2. Let $\mathscr{A}$ be an adversary against the Φ_d-statistical-key-collision security that makes $Q_{\mathscr{A}}$ queries. Since the function F defined in the **Initialize** procedure is a random function, $\mathscr{A}$ does not learn any information on the key $\vec{a}$ until $b' \leftarrow 1$, so $\mathbf{Adv}^{\mathsf{skc}}_{\Phi_d}(\mathscr{A})$ is bounded by the probability that $\mathscr{A}$ makes use in its queries of two different RKD functions that lead to the same key. We claim that

$$\mathbf{Adv}^{\mathsf{skc}}_{\Phi_d}(\mathscr{A}) \ \leq \ \frac{d \cdot Q_{\mathscr{A}}^2}{2p} \ .$$

This follows easily on noting that, if two different RKD-functions do lead to the same key, then those two functions must differ in some coordinate k, meaning that the difference in those components is a non-zero polynomial ψ_k of degree at most d such that $\psi_k(a_k) = 0$. Here a_k is the k-th component of vector $\vec{a}$ that was selected uniformly at random in the **Initialize** procedure. Since ψ_k has at most d roots and a_k is uniformly random in $\mathbb{Z}_p$, the probability that $\psi_k(a_k) = 0$ is bounded by d/p. To obtain the final result, one simply applies a union bound over the (at most) $\binom{Q_{\mathscr{A}}}{2}$ pairs of choices of different RKD-functions accessed by $\mathscr{A}$. □

Lemma 6.3. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let NR^* be defined via $\mathsf{NR}^*(\vec{a}, x) = [\prod_{i=1}^{n} a_i^{x_i}]$, where $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. Let $\mathcal{S}$ be the set $\{0,1\}^n \setminus (\{0^n\} \cup \{\omega_1, \ldots, \omega_n\})$. Let $\mathscr{A}$ be an adversary against the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^* that makes $Q_{\mathscr{A}}$ oracle queries. Then, assuming $nd \leq \sqrt{p}$, we can design an adversary $\mathscr{B}$ against the d-DDHI problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}^{(\mathcal{S},\Phi_d)\text{-ui-prf-rka}}_{\mathsf{NR}^*}(\mathscr{B}) \leq \left(n \cdot d \cdot \left(\frac{p}{p-1} \right)^2 + n \cdot (d-1) \right) \cdot \mathbf{Adv}^{d\text{-ddhi}}_{\mathbb{G}}(\mathscr{A}) + \frac{2n \cdot Q_{\mathscr{A}}}{p}.$$

The running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time required to compute $O(d \cdot (n + Q_{\mathscr{A}}))$ exponentiations in $\mathbb{G}$ and $O(Q_{\mathscr{A}}^3 \cdot (nd + Q_{\mathscr{A}}))$ operations in $\mathbb{Z}_p$.

Proof of Lemma 6.3. In order to prove Lemma 6.3, let us first introduce the following intermediate problems, which are related to the d-DDHI problem in $\mathbb{G}$.

(N,d)-Polynomial DDH ((N,d)-PDDH). Let $\mathbb{G}$ be a group of prime order p. For $d \geq 1$, the (N,d)-PDDH problem in $\mathbb{G}$ consists of deciding, given $([1], \vec{X}_1, \ldots, \vec{X}_N)$ with $\mathbb{G} = \langle g \rangle$ and $\vec{X}_i = ([a_i], \ldots, [a_i^d], z_i)$, where $a_i \in \mathbb{Z}_p^*$, for $i = 1, \ldots, N$, whether $z_i = [a_i^{d+1}]$ for $i = 1, \ldots, N$ (corresponding to (N,d)-PDDH-Real) or whether $z_i = [c_i]$ for random $c_i \in \mathbb{Z}_p^*$ for $i = 1, \ldots, N$ (corresponding to (N,d)-PDDH-Rand). The advantage of an adversary $\mathscr{B}$ against the (N,d)-PDDH problem in $\mathbb{G}$, denoted by $\mathbf{Adv}_d^{(Adv,N)\text{-}\mathsf{pddh}}(\mathbb{G})\mathscr{B}$ is defined to be:

$$\mathbf{Adv}_{\mathbb{G}}^{(N,d)\text{-}\mathsf{pddh}}(\mathscr{B}) = \Pr[\,(N,d)\text{-PDDH-Real} \Rightarrow 1\,] - \Pr[\,(N,d)\text{-PDDH-Rand} \Rightarrow 1\,]$$

where the probabilities are over $a_i, c_i \xleftarrow{\$} \mathbb{Z}_p^*$, $i = 1, \ldots, N$.

(N,d)-Extended DDH ((N,d)-EDDH). Let $\mathbb{G}$ be a group of prime order p with generator $[1]$. Then the (N,d)-EDDH problem in $\mathbb{G}$ consists of deciding, given $\vec{Z} = (Z_{k,l})$, for $k = 0, \ldots, N$ and $l = 0, \ldots, d$ with $Z_{0,0} = [1]$, whether $Z_{k,l} = [a_k b^l]$, with $a_0 = 1$, $a_k \in \mathbb{Z}_p^*$ for $k = 1, \ldots, N$ and $b \in \mathbb{Z}_p^*$ (corresponding to (N,d)-EDDH-Real) or whether $Z_{k,l} = [c_{k,l}]$, for $(k,l) \neq (0,0)$, with $c_{k,l} \in \mathbb{Z}_p^*$ (corresponding to (N,d)-EDDH-Rand). The advantage of an adversary $\mathscr{B}$ against the (N,d)-EDDH problem in $\mathbb{G}$, denoted by $\mathbf{Adv}_{\mathbb{G}}^{(N,d)\text{-}\mathsf{eddh}}(\mathscr{B})$ is defined to be:

$$\mathbf{Adv}_{\mathbb{G}}^{(N,d)\text{-}\mathsf{eddh}}(\mathscr{B}) = \Pr[\,(N,d)\text{-EDDH-Real} \Rightarrow 1\,] - \Pr[\,(N,d)\text{-EDDH-Rand} \Rightarrow 1\,]$$

where the probabilities are over $a_k \xleftarrow{\$} \mathbb{Z}_p^*$, for $k = 1, \ldots, N$, $b \xleftarrow{\$} \mathbb{Z}_p^*$, and $c_{k,l} \xleftarrow{\$} \mathbb{Z}_p^*$, for $(k,l) \neq (0,0)$.

Let $\mathcal{S}$ denote $\{0,1\}^n \setminus (\{\omega_1, \ldots, \omega_n\} \cup \{0^n\})$. To prove the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^* based on the d-DDHI problem in $\mathbb{G}$, we first prove a similar statement in Lemma 6.4 based on hardness of the (N,d)-PDDH and the (N,d)-EDDH problems in $\mathbb{G}$. Then, in Lemmas 6.5–6.10, we relate the hardness of both these problems to the hardness of the d-DDHI problem in $\mathbb{G}$.

Lemma 6.4. *Let $\mathbb{G}$ be a group of prime order p. Let $g = [1]$ be a generator of $\mathbb{G}$ and NR^*: $\mathbb{Z}_p^n \times (\{0,1\}^n \setminus \{0^n\}) \to \mathbb{G}$ defined via $\mathsf{NR}^*(\vec{a}, x) = [\prod_{i=1}^n a_i^{x_i}]$. Let $\mathcal{S}$ denote the set $\{0,1\}^n \setminus (\{0^n\} \cup \{\omega_1, \ldots, \omega_n\})$. Let $\mathscr{B}$ be an adversary against the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^* that makes $Q_{\mathscr{B}}$ oracle queries. Then, assuming $nd \leq \sqrt{p}$, we can design adversaries $\mathscr{B}_j$ against the $(Q_{\mathscr{B}}, d)$-EDDH problem in $\mathbb{G}$, for $j = 0, \ldots, n-1$, and adversaries $\mathscr{D}_k$ against the (n,k)-PDDH problem in $\mathbb{G}$, for $k = 1, \ldots, d-1$ such that*

$$\mathbf{Adv}_{\mathsf{NR}^*}^{(\mathcal{S},\Phi_d)\text{-}\mathsf{ui\text{-}prf\text{-}rka}}(\mathscr{B}) \leq \sum_{j=0}^{n-1} \mathbf{Adv}_{\mathbb{G}}^{(Q_{\mathscr{B}},d)\text{-}\mathsf{eddh}}(\mathscr{B}_j) + \frac{2n \cdot Q_{\mathscr{B}}}{p} + \sum_{k=1}^{d-1} \mathbf{Adv}_{\mathbb{G}}^{(n,k)\text{-}\mathsf{pddh}}(\mathscr{D}_k)\,. \quad (3)$$

The running time of $\mathscr{B}_j$ is that of $\mathscr{B}$ plus $O(Q_{\mathscr{B}}^3(n \cdot d + Q_{\mathscr{B}}))$ operations in $\mathbb{Z}_p$. The running time of $\mathscr{D}_k$ is that of $\mathscr{B}$ plus the time required to compute (at most) $d \cdot Q_{\mathscr{B}}$ exponentiations in $\mathbb{G}$.

Proof of Lemma 6.4. The proof is based on the sequence of games of Figure 7. Let $\mathscr{B}$ be an adversary against the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^*, so $\mathscr{B}$ never queries the same entry x twice, for any $x \in \mathcal{S}$.

Preliminaries. For a RKD-function $\vec{\phi} = (\phi_1, \ldots, \phi_n) \in \Phi_d$, we let ϕ_i be the polynomial defined by $\phi_i : T_i \mapsto \sum_{k=0}^d \alpha_{i,k} \cdot T_i^k$, for each $i = 1, \ldots, n$.

proc Initialize // G_0, G_2
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$; Φ_d

proc RKFn$(\vec{\phi}, x)$ // G_0
$y \leftarrow \mathsf{NR}^*(\vec{\phi}(\vec{a}), x)$
Return y

proc Finalize(b') // G_0–G_2
Return b

proc Initialize // G_1
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$; Φ_d
$\mathsf{T}[0^{i-1} \,\|\, l \,\|\, 0^{n-i}] \xleftarrow{\$} \mathbb{G}$,
  $\forall l = 1, \ldots, d, \forall i = 1, \ldots, n$
$\mathsf{T}[0^n] \leftarrow [1]$

proc RKFn$(\vec{\phi}, x)$ // G_1
// $\vec{\phi} = (\phi_1, \ldots, \phi_n)$
// $\phi_i : \vec{T} \mapsto \sum_{j=0}^{d} \alpha_{i,j} \cdot T_i^j$
// $\forall i = 1, \ldots, n$
If $x \in \mathcal{S}$ then
  $y \xleftarrow{\$} \mathbb{G}$
Else // $x = \omega_i$ for $i \in \{1, \ldots, n\}$
  $y \leftarrow \prod_{k=0}^{d} \mathsf{T}[0^{i-1} \,\|\, k \,\|\, 0^{n-i}]^{\alpha_{i,k}}$
Return y

proc RKFn$(\vec{\phi}, x)$ // G_2
If $x \in \mathcal{S}$ then
  $y \xleftarrow{\$} \mathbb{G}$
Else
  $y \leftarrow \mathsf{NR}^*(\vec{\phi}(\vec{a}), x)$
Return y

Figure 7: Game for the proof of the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^*.

To each query $(\vec{\phi}, x)$ of the adversary, we associate the following polynomials, for $j = 1, \ldots, n$:

$$P_{\vec{\phi},x,j}(\vec{T}) = \prod_{i=1}^{j} \phi_i(T_i)^{x_i} = \sum_{z \preceq d \cdot x_{1,\ldots,j}} \prod_{i=1}^{j} \alpha_{i,z_i} T_i^{z_i} \ ,$$

with indeterminates $\vec{T} = (T_1, \ldots, T_j)$. These polynomials may have up to $(d+1)^j$ (distinct) monomials and so cannot be expanded efficiently. But they can still be formally considered as row vectors $(P^{(z)}_{\vec{\phi},x,j})_{z \in \{0,\ldots,d\}^j}$ in $\mathbb{Z}_p^{(d+1)^j}$, where $P^{(z)}_{\vec{\phi},x,j}$ is the coefficient of the monomial $T_1^{z[1]} \cdots T_j^{z[j]}$.

As vectors, they can be multiplied to other vectors or matrices (with indices from the set $\{0, \ldots, d\}^j$) over $\mathbb{Z}_p$. We can also define the multiplication of such a vector with a column vector over $\mathbb{G}$. Specifically, if $\vec{U} = (U_z)_z$ is a column vector with entries from $\mathbb{G}$, then we write:

$$P_{\vec{\phi},x,j} \odot \vec{U} = \prod_{z \in \{0,\ldots,d\}^j} U_z^{P^{(z)}_{\vec{\phi},x,j}} = \prod_{z \preceq d \cdot x_{1,\ldots,j}} U_z^{\prod_{i=1}^{j} \alpha_{i,z_i}} \ .$$

Let us suppose that we have a polynomial-time procedure **TestLin** which takes as input j, a list $\mathcal{L}$ of pairs $(\vec{\phi}_l, x_l)$ (for $l = 1, \ldots, L$, such that $P_{\vec{\phi}_l,x_l,j}$ are linearly independent as polynomials) together with a pair $(\vec{\phi}, x)$ and which outputs:

$$\begin{cases} \perp & \text{if } P_{\vec{\phi},x,j} \text{ is linearly independent of the set } \{P_{\vec{\phi}_l,x_l,j} | l = 1, \ldots, L\} \\ \vec{\lambda} = (\lambda_1, \ldots, \lambda_L) & \text{otherwise, so that } P_{\vec{\phi},x,j} = \sum_{l=1}^{L} \lambda_l P_{\vec{\phi}_l,x_l,j} \end{cases}$$

Since the $P_{\vec{\phi}_l,x_l,j}$ are linearly independent polynomials, there is at most one possible $\vec{\lambda}$. Unfortunately, we do not know any such polynomial-time procedure. But, as we will see later, we can approximate such a procedure by evaluating the polynomials, and this is sufficient for our purposes.

Indistinguishability of Game G_0 and Game G_1. It is clear that game G_0 instantiates exactly the game defining the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^* when $b = 0$.

In game G_1, we respond to queries in $\mathcal{S}$ by uniformly random values, as is done in the game defining the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^* when $b = 1$. However we do not reply to queries $x \notin \mathcal{S}$ as is done in that game. We design adversaries $\mathscr{B}_j$ attacking the $(Q_{\mathscr{B}}, d)$-EDDH problem in $\mathbb{G}$ such that

$$\Pr[\,\textsc{Succ}_0\,] - \Pr[\,\textsc{Succ}_1\,] \;\leq\; \sum_{j=0}^{n-1} \mathbf{Adv}_{\mathbb{G}}^{(Q_{\mathscr{B}},d)\text{-eddh}}(\mathscr{B}_j)\;.$$

For that purpose, we use the sequence of games in Figure 8, in the following order: $\mathrm{G}_{0,0}, \mathrm{G}'_{0,0}$, $\mathrm{G}_{0,1}, \ldots, \mathrm{G}'_{0,n-1}, \mathrm{G}_{0,n}$. More precisely, we prove that $\mathrm{G}_{0,j}$ is indistinguishable from $\mathrm{G}'_{0,j}$ under the $(Q_{\mathscr{B}}, d)$-EDDH assumption, while we show that $\mathrm{G}'_{0,j}$ is perfectly indistinguishable from $\mathrm{G}_{0,j+1}$.

In $\mathrm{G}_{0,0}$, **TestLin** always returns an empty vector $\vec{\lambda}$, $\mathcal{L}$ and T remain empty, and y is set to 1, i.e., the empty product. So $\mathrm{G}_{0,0}$ is exactly G_0.

proc Initialize // $\mathrm{G}_{0,j}$; $j = 0, \ldots, n$
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$
$\mathcal{L} \leftarrow []$
$\mathsf{T} \leftarrow []$
$L \leftarrow 0$
proc RKFn$(\vec{\phi}, x)$ // $\mathrm{G}_{0,j}$; $j = 0, \ldots, n$
// $\vec{\phi} = (\phi_1, \ldots, \phi_n)$
// $\phi_i : \vec{T} \mapsto \sum_{j=0}^{d} \alpha_{i,j} \cdot T_i^j$, $\forall i = 1, \ldots, n$
$\vec{\lambda} \leftarrow$ **TestLin**$(j, \mathcal{L}, (\vec{\phi}, x))$
If $\vec{\lambda} = \perp$ then
  $L \leftarrow L + 1$
  $\mathcal{L}[L] \leftarrow (\vec{\phi}, x)$
  $\mathsf{T}[L] \xleftarrow{\$} \mathbb{G}$
  $\vec{\lambda} \leftarrow (0, \ldots, 0, 1) \in \mathbb{Z}_p^L$
$y \leftarrow \prod_{l=1}^{L} \mathsf{T}[l]^{\lambda_l}$
$y' \leftarrow y^{\prod_{i=j+1}^{n} \phi_i(a_i)^{x_i}}$
Return y'

proc Initialize // $\mathrm{G}'_{0,j}$; $j = 0, \ldots, n-1$
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$
$\mathcal{L} \leftarrow []$
$\mathsf{T} \leftarrow []$
$L \leftarrow 0$
proc RKFn$(\vec{\phi}, x)$ // $\mathrm{G}_{0,j}$; $j = 0, \ldots, n$
// $\vec{\phi} = (\phi_1, \ldots, \phi_n)$
// $\phi_i : \vec{T} \mapsto \sum_{j=0}^{d} \alpha_{i,j} \cdot T_i^j$, $\forall i = 1, \ldots, n$
$\vec{\lambda} \leftarrow$ **TestLin**$(j, \mathcal{L}, (\vec{\phi}, x))$
If $\vec{\lambda} = \perp$ then
  $L \leftarrow L + 1$
  $\mathcal{L}[L] \leftarrow (\vec{\phi}, x)$
  For $k = 0, \ldots, d$
    $\mathsf{T}[L, k] \xleftarrow{\$} \mathbb{G}$
  $\vec{\lambda} \leftarrow (0, \ldots, 0, 1) \in \mathbb{Z}_p^L$
$y \leftarrow \prod_{l=1}^{L} \prod_{k=0}^{d \cdot x_{j+1}} \mathsf{T}[l, k]^{\lambda_l \cdot \alpha_{j+1,k}^{x_{j+1}}}$
$y' \leftarrow y^{\prod_{i=j+2}^{n} \phi_i(a_i)^{x_i}}$
Return y'

Figure 8: Games $\mathrm{G}_{0,j}$ and $\mathrm{G}'_{0,j}$ for the proof of Lemma 6.4.

Indistinguishability of Game $\mathrm{G}_{0,j}$ and Game $\mathrm{G}'_{0,j}$ under the $(Q_{\mathscr{B}}, d)$-EDDH assumption.
Let us now design adversaries $\mathscr{B}_j$ attacking the $(Q_{\mathscr{B}}, d)$-EDDH problem in $\mathbb{G}$ such that

$$\Pr[\,\textsc{Succ}_{0,j}\,] - \Pr\left[\,\textsc{Succ}'_{0,j}\,\right] \;\leq\; \mathbf{Adv}_{\mathbb{G}}^{(Q_{\mathscr{B}},d)\text{-eddh}}(\mathscr{B}_j)\;;\; \forall j = 0, \ldots, n-1$$

assuming the existence of a perfect **TestLin** oracle (which we recall does not exist). Later, we will get the real bound using a concrete, approximate **TestLin** procedure.

Let $\vec{Z}$ be an (N, d)-EDDH tuple. So $Z_{0,0} = [1] = g$ with $\mathbb{G} = \langle g \rangle$. In the $(Q_{\mathscr{B}}, d)$-EDDH-Real case, we have $Z_{l,k} = \left[a_l b^k\right]$, with $a_0 = 1$ and $b, a_l \xleftarrow{\$} \mathbb{Z}_p^*$ for $l = 1, \ldots, Q_{\mathscr{B}}$ and $k = 0, \ldots, d$. In the $(Q_{\mathscr{B}}, d)$-EDDH-Rand case, we have $Z_{l,k} = [c_{l,k}]$ where $c_{l,k} \xleftarrow{\$} \mathbb{Z}_p^*$, for $(l, k) \in \{0, \ldots, Q_{\mathscr{B}}\} \times \{0, \ldots, d\}$, $(l, k) \neq (0, 0)$.

Adversary $\mathscr{B}_j$ starts by picking $a_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = j+2, \ldots, n$. Adversary $\mathscr{B}_j$ then runs $\mathscr{A}$. When the latter makes an **RKFn**-query $(\vec{\phi}, x)$, adversary $\mathscr{B}_j$ does everything as in Game $\mathrm{G}_{0,j}$, except it does not return y' but instead computes z and z' as follows:

$$z \leftarrow \prod_{l=1}^{L} \prod_{k=0}^{d \cdot x_{j+1}} Z_{l,k}^{\lambda_l \cdot \alpha_{j+1,k}^{x_{j+1}}}$$

$$z' \leftarrow z^{\prod_{i=j+2}^{n} \phi_i(a_i)^{x_i}}$$

and returns z'. Adversary $\mathscr{B}_j$ does not compute y and y'.

If $\vec{Z}$ is a real $(Q_{\mathscr{B}}, d)$-EDDH tuple, then

$$z = \prod_{l=1}^{L} \prod_{k=0}^{d \cdot x_{j+1}} Z_{l,0}^{b^k \cdot \lambda_l \cdot \alpha_{j+1,k}^{x_{j+1}}} = \prod_{l=1}^{L} Z_{l,0}^{\lambda_l \cdot \phi_{j+1}(b)^{x_{j+1}}} = \left(\prod_{l=1}^{L} Z_{l,0}^{\lambda_l} \right)^{\phi_{j+1}(b)^{x_{j+1}}} ,$$

and so $z = y^{\phi_{j+1}(b)}$ and $z' = y'$, where y and y' are computed as in Game $\mathrm{G}_{0,j}$ and when $a_j = b$ and $\mathsf{T}[l] = Z_{l,0}$ (which are random and used nowhere else). So, in this case, $\mathscr{B}_j$ simulates perfectly Game $\mathrm{G}_{0,j}$.

Now, if $\vec{Z}$ is a random $(Q_{\mathscr{B}}, d)$-EDDH tuple, then $\mathscr{B}_j$ simulates perfectly Game $\mathrm{G}_{0,j}$, when $\mathsf{T}[l,k] = Z_{l,k}$ (which are random and used nowhere else), $z' = y'$ and $z = y$.

Perfect Indistinguishability of Game $\mathrm{G}'_{0,j}$ and Game $\mathrm{G}_{0,j+1}$. It remains to prove that Game $\mathrm{G}'_{0,j}$ is perfectly indistinguishable from Game $\mathrm{G}_{0,j+1}$. To establish that, we will consider another intermediate game (Game $\mathrm{G}''_{0,j}$ in Figure 9). This game is not polynomial-time (since $\vec{U}$ contains $(d+1)^{j+1}$ entries), but we will show that it is perfectly indistinguishable from Game $\mathrm{G}'_{0,j}$ and Game $\mathrm{G}_{0,j+1}$.

proc Initialize // $\mathrm{G}''_{0,j}$; $j = 0, \ldots, n-1$	**proc RKFn**$(\vec{\phi}, x)$ // $\mathrm{G}''_{0,j}$; $j = 0 \ldots, n-1$
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$	// $\vec{\phi} = (\phi_1, \ldots, \phi_n)$
$U_x \xleftarrow{\$} \mathbb{G}$; $\forall x \in \{0, \ldots, d\}^{j+1} \setminus \{0^{j+1}\}$	// $\phi_i : T_i \mapsto \sum_{j=0}^{d} \alpha_{i,j} \cdot T_i^j$, $\forall i = 1, \ldots, n$
$U_{0^{j+1}} \leftarrow [1]$	$y \leftarrow P_{\vec{\phi},x,j+1} \odot \vec{U}$
	$y' \leftarrow y^{\prod_{i=j+2}^{n} \phi_i(a_i)^{x_i}}$
	Return y

Figure 9: Games $\mathrm{G}''_{0,j}$ for the proof of Lemma 6.4.

Let us first prove that Game $\mathrm{G}'_{0,j}$ is perfectly indistinguishable from Game $\mathrm{G}''_{0,j}$. We do this in two steps:

1. Let us show that we can compute T in $\mathrm{G}'_{0,j}$ as follows:

$$\mathsf{T}[l,k] = \left(P_{\vec{\phi}_l,x_l,j} \cdot T_{j+1}^k \right) \odot \vec{U} \tag{4}$$

with $\vec{U}$ computed as in $\mathrm{G}''_{0,j}$ and $\mathcal{L}[l] = (\vec{\phi}_l, x_l)$. If we look at $\mathsf{T} = (\mathsf{T}[l,k])_{l,k}$ as a vector over $\mathbb{Z}_p^{L(d+1)}$ and $\boldsymbol{M} = (P_{\vec{\phi}_l,x_l,j} \cdot T_{j+1}^k)_{l,k}$ as a matrix of $L(d+1)$ rows and $(d+1)^{j+1}$ columns (each row corresponding to a polynomial $P_{\vec{\phi}_l,x_l,j} \cdot T_{j+1}^k$), then we can write this as:

$$\mathsf{T} = \boldsymbol{M} \odot \vec{U} .$$

But since the polynomials $P_{\vec{\phi}_l,x_l,j}$ are linearly independent (and do not contain T_{j+1}), the rows of $\boldsymbol{M}$ are also linearly independent, and $\boldsymbol{M}$ is full rank. Therefore, if $\vec{U}$ is random, then so is T, exactly as in Game $\mathrm{G}'_{0,j}$.

2. Supposing T is computed as in Equation (4), then the output y in Game $\mathrm{G}'_{0,j}$ is equal to:

$$y = \prod_{l=1}^{L}\prod_{k=0}^{d\cdot x_{j+1}} \mathsf{T}[l,k]^{\lambda_l\cdot\alpha_{j+1,k}^{x_{j+1}}} = \prod_{l=1}^{L}\prod_{k=0}^{d\cdot x_{j+1}} \left(\left(\lambda_l\cdot\alpha_{j+1,k}^{x_{j+1}}\cdot P_{\vec{\phi}_l,x_l,j}\cdot T_{j+1}^{k}\right)\odot\vec{U}\right)$$
$$= \left(\sum_{l=1}^{L}\lambda_l\cdot P_{\vec{\phi}_l,x_l,j}\cdot\phi_{j+1}(T_{j+1})^{x_{j+1}}\right)\odot\vec{U} = \left(P_{\vec{\phi},x,j}\cdot\phi_{j+1}(T_{j+1})^{x_{j+1}}\right)\odot\vec{U} = P_{\vec{\phi},x,j+1}\odot\vec{U}$$

which is exactly the way it is computed in Game $\mathrm{G}''_{0,j}$.

Let us now prove that Game $\mathrm{G}''_{0,j}$ is perfectly indistinguishable from Game $\mathrm{G}_{0,j+1}$. We again use two steps, which are very similar to the previous ones:

1. Let us show that we can compute T in $\mathrm{G}_{0,j+1}$ as follows:

$$\mathsf{T}[l] = P_{\vec{\phi}_l,x_l,j+1}\odot\vec{U} \tag{5}$$

with $\vec{U}$ computed as in $\mathrm{G}''_{0,j}$ and $\mathcal{L}[l] = (\vec{\phi}_l, x_l)$. If we look at $\mathsf{T} = (\mathsf{T}[l])_l$ as a vector over $\mathbb{Z}_p^L$ and $\boldsymbol{M} = (P_{\vec{\phi}_l,x_l,j+1})_l$ as a matrix of L rows and $(d+1)^{j+1}$ columns (each row corresponding to a polynomial $P_{\vec{\phi}_l,x_l,j+1}$), then we can write this as:

$$\mathsf{T} = \boldsymbol{M}\odot\vec{U}\ .$$

But since the polynomials $P_{\vec{\phi}_l,x_l,j+1}$ are linearly independent, the rows of $\boldsymbol{M}$ are also linearly independent, and $\boldsymbol{M}$ is full rank. Therefore, if $\vec{U}$ is random, then so is T, exactly as in Game $\mathrm{G}_{0,j+1}$.

2. Supposing T is computed as in Equation (5), then the output y in Game $\mathrm{G}_{0,j+1}$ is equal to:

$$y = \prod_{l=1}^{L}\mathsf{T}[l]^{\lambda_l} = \prod_{l=1}^{L}\left(\left(\lambda_l\cdot P_{\vec{\phi}_l,x_l,j+1}\right)\odot\vec{U}\right)$$
$$= \left(\sum_{l=1}^{L}\lambda_l\cdot P_{\vec{\phi}_l,x_l,j+1}\right)\odot\vec{U} = P_{\vec{\phi},x,j+1}\odot\vec{U}$$

which is exactly the way it is computed in Game $\mathrm{G}''_{0,j}$.

Perfect Indistinguishability of Game $\mathrm{G}_{0,n}$ and Game G_1. Let us now prove that Game $\mathrm{G}_{0,n}$ is perfectly indistinguishable from Game G_1. We just need to prove that all polynomials $P_{\vec{\phi},x,n}$ corresponding to queries $(\vec{\phi},x)$ with $x\in\mathcal{S}$ are linearly independent of all other polynomials $P_{\vec{\phi}',x',n}$ corresponding to queries $(\vec{\phi}',x')$ with $x'\neq x$. To prove this, let us suppose $P_{\vec{\phi},x,n}$ with $x\in\mathcal{S}$ is a linear combination of some $P_{\vec{\phi}_2,x_2,n},\ldots,P_{\vec{\phi}_m,x_m,n}$:

$$\sum_{i=1}^{m}\lambda_i\cdot P_{\vec{\phi}_i,x_i,n} = 0\ ,$$

with $\lambda_i \neq 0$ for all i, and with $\vec{\phi}_1 = \vec{\phi}$ and $x_1 = x$. Then, in this sum, let us consider an arbitrary monomial $T_1^{z_1} \cdots T_n^{z_n}$ with z of highest Hamming weight. Necessarily, the Hamming weight of z is at least 2, since the Hamming weight of $x_1 = x \in \mathcal{S}$ is at least 2. But, since the sum is the zero polynomial, there must exist two distinct polynomials $P_{\vec{\phi}_i,x_i,n}$ and $P_{\vec{\phi}_{ij},x_j,n}$ containing this monomial $T_1^{z_1} \cdots T_n^{z_n}$.

Let $\hat{z}$ be the n-bit string such that $\hat{z}_i = 0$ if $z_i = 0$, while $\hat{z}_i = 1$ otherwise, for all i. Then, since z has the highest possible Hamming weight, $x_i = x_j = \hat{z}$ (from the definitions of $P_{\vec{\phi}_i,x_i,n}$ and $P_{\vec{\phi}_j,x_j,n}$). In addition $\hat{z} \in \mathcal{S}$, because the Hamming weight of z is at least 2, and so is the Hamming weight of $\hat{z}$. This means the adversary $\mathscr{B}$ queried twice $\hat{z} \in \mathcal{S}$, which is forbidden.

TestLin Procedure. It remains to provide a polynomial-time **TestLin** procedure. Unfortunately, we do not know any polynomial-time exact procedure, but we provide an approximate one in Figure 10. We assume in the analysis that $nd \leq \sqrt{p}$, which is true for sufficiently large p and fixed d, n.

Let us prove that this approximate procedure is incorrect with probability at most $\frac{1}{p}$ (over its random coins). The polynomials $P_{\vec{\phi}_l,x_l,j}$ with $l = 1, \ldots, L$ are supposed to be linearly independent. Then, there are two cases:

1. If $P_{\vec{\phi},x,j} = P_{\vec{\phi}_{L+1},x_{L+1},j}$ is linearly independent from $P_{\vec{\phi}_1,x_1,j}, \ldots, P_{\vec{\phi}_L,x_L,j}$, then the probability that the procedure does not return $\perp$ is (over the value of X):

$$\Pr\left[\exists \vec{\lambda} \in \mathbb{Z}_p^{(L+1)},\ \vec{\lambda} \cdot \boldsymbol{M} = 0\right] \leq \sum_{\vec{\lambda} \in \mathbb{Z}_p^{(L+1)}} \Pr\left[\vec{\lambda} \cdot \boldsymbol{M} = 0\right]$$
$$\leq \sum_{\vec{\lambda} \in \mathbb{Z}_p^{(L+1)}} \Pr\left[\forall k = 1, \ldots, N,\ (\sum_{l=1}^{L+1} \lambda_l P_{\vec{\phi}_l,x_l,j})(\gamma_k) = 0\right]$$

   and $\sum_{l=1}^{L+1} \lambda_l P_{\vec{\phi}_l,x_l,j}$ is a non-zero polynomial of degree at most jd. Since γ_k are chosen independently and uniformly at random in $\mathbb{Z}_p^n$, according to the Schwartz-Zippel lemma, the error probability is at most:

$$\sum_{\vec{\lambda} \in \mathbb{Z}_p^{(L+1)}} \left(\frac{jd}{p}\right)^N = p^{L+1} \cdot \left(\frac{jd}{p}\right)^N \leq p^{L+1} \cdot \frac{1}{p^{L+2}} = \frac{1}{p}\,,$$

   since $jd \leq nd \leq \sqrt{p}$ and $N = 2L + 4$.

2. If $P_{\vec{\phi},x,j} = P_{\vec{\phi}_{l+1},x_{L+1},j}$ is such that there exists $\vec{\lambda} \in \mathbb{Z}_p^L$ such that $P_{\vec{\phi},x,j} = \sum_{l=1}^{L} \lambda_l P_{\vec{\phi}_l,x_l,j}$, then such $\vec{\lambda}$ is unique. Let us prove that the probability that the **TestLin** procedure does not return $\vec{\lambda}$ is at most $\frac{1}{p}$. Let Λ be the set of $\vec{\lambda'} \in \mathbb{Z}_p^{L+1}$ such that $\lambda'_{L+1} \cdot \vec{\lambda} \neq \vec{\lambda'}_{1,\ldots,L}$. Then the error probability of the **TestLin** procedure is at most:

$$\Pr\left[\exists \vec{\lambda'} \in \Lambda,\ \vec{\lambda'} \cdot \boldsymbol{M} = 0\right] \leq \sum_{\vec{\lambda'} \in \Lambda} \Pr\left[\forall k = 1, \ldots, N,\ (\sum_{l=1}^{L+1} \lambda'_l P_{\vec{\phi}_l,x_l,j})(\gamma_k) = 0\right].$$

   Moreover, $\sum_{l=1}^{L+1} \lambda'_l P_{\vec{\phi}_l,x_l,j}$ is a polynomial of degree at most jd, which is non-zero because otherwise the $P_{\vec{\phi}_1,x_1,j}, \ldots, P_{\vec{\phi}_L,x_L,j}$ would not be independent. We can conclude the proof as in the first case, since $|\Lambda| \leq |\mathbb{Z}_p^{(L+1)}|$.

procTestLin$(j, \mathcal{L}, (\vec{\phi}, x))$

// $\mathcal{L}[l] = (\vec{\phi}_l, x_l)$ for $l = 1, \ldots, L$ and $L = |\mathcal{L}|$

$(\vec{\phi}_{L+1}, x_{L+1}) \leftarrow (\vec{\phi}, x)$

$N \leftarrow 2L + 4$

$\boldsymbol{M}$ matrix over $\mathbb{Z}_p$ of $L + 1$ rows and N columns

X table of $L + 3$ vectors in $\mathbb{Z}_p^n$

For $k = 1, \ldots, N$

  $\gamma_k \xleftarrow{\$} \mathbb{Z}_p^n$

For $i = 1, \ldots, L + 1$

  For $k = 1, \ldots, N$

    $M_{i,k} \leftarrow P_{\vec{\phi}_i, x_i}(\gamma_k)$

Apply Gaussian elimination on $\boldsymbol{M}$

If $\boldsymbol{M}$ is full-rank then

  Return $\perp$

Else

  Let $\vec{\lambda}'$ be the row vector such that $\vec{\lambda}' \cdot \boldsymbol{M} = \vec{0}$

  $\vec{\lambda} \leftarrow (\lambda'_1/\lambda'_{L+1}, \ldots, \lambda'_L/\lambda'_{L+1})$

  Return $\vec{\lambda}$

Figure 10: **TestLin** procedure.

Therefore replacing a perfect **TestLin** oracle by this **TestLin** procedure is $1/p$-statistically indistinguishable. In addition, the only parts where we supposed **TestLin** to be exact, was for proving the perfect indistinguishability of $\mathrm{G}'_{0,j}$ and $\mathrm{G}_{0,j+1}$ (for $j = 0, \ldots, n-2$), and the perfect indistinguishability of $\mathrm{G}'_{0,n-1}$, $\mathrm{G}_{0,n}$ and G_1 (note that we actually do not need $\mathrm{G}_{0,n}$ and we can directly go from $\mathrm{G}'_{0,n-1}$ to G_1). The computationnal indistinguishability of $\mathrm{G}_{0,j}$ and $\mathrm{G}'_{0,j}$ does not use any property of **TestLin**. Since this procedure is called at most $Q_{\mathscr{B}}$ times in $\mathrm{G}_{0,j}$, $\mathrm{G}_{0,j'}$, and G_1 we have:

$$\Pr\left[\,\textsc{Succ}'_{0,j}\,\right] - \Pr\left[\,\textsc{Succ}_{0,j+1}\,\right] \;\leq\; \frac{Q_{\mathscr{B}}}{p}\;;\; \forall j = 0, \ldots, n-2\;,$$

$$\Pr\left[\,\textsc{Succ}'_{0,n-1}\,\right] - \Pr\left[\,\textsc{Succ}_{1}\,\right] \;\leq\; \frac{Q_{\mathscr{B}}}{p}\;.$$

TestLin evaluates $L + 1$ polynomials (which are themselves products of j univariate polynomials of degree d) in $N = 2L + 4$ points, which costs $O(LNdj) = O(L^2dn)$ operations in $\mathbb{Z}_p$ (using the Hörner scheme); and then it does a Gaussian elimination on a matrix of $L + 1$ rows and N columns, which costs $O(N^3) = O(L^3)$ operations in $\mathbb{Z}_p$. In total, $\mathscr{B}_j$ has a running time that is the same as that of $\mathscr{B}$ plus $O(Q_{\mathscr{B}}^3(dn + Q_{\mathscr{B}}))$ operations in $\mathbb{Z}_p$ (since $L \leq Q_{\mathscr{B}}$).

Hence, we can prove that:

$$\Pr\left[\,\textsc{Succ}_0\,\right] - \Pr\left[\,\textsc{Succ}_1\,\right] \;\leq\; \sum_{j=0}^{n-1} \mathbf{Adv}_{\mathbb{G}}^{(Q_{\mathscr{B}},d)\text{-}\mathsf{eddh}}(\mathscr{B}_j) + \frac{2nQ_{\mathscr{B}}}{p}\;. \tag{6}$$

Indistinguishability of Game G_1 **and Game** G_2 **under** (n, k)**-PDDH.** Game G_2 instantiates exactly the game defining the $(\mathcal{S}, \Phi_d)$-unique-input-prf-rka security of NR^* when $b = 1$, so the only difference with game G_1 is in the way queries x with $x \notin \mathcal{S}$ are handled. We design adversaries $\mathscr{D}_k$

against the (n,k)-PDDH problem in $\mathbb{G}$ such that

$$\Pr\left[\,\textsc{Succ}_1\,\right] - \Pr\left[\,\textsc{Succ}_2\,\right] \;\leq\; \sum_{k=1}^{d-1} \mathbf{Adv}_{\mathbb{G}}^{(n,k)\text{-}\mathsf{pddh}}(\mathscr{D}_k)\;. \tag{7}$$

We prove this statement using the sequence of games of Figure 11.

proc Initialize // $\mathrm{G}_{1,k}$; $k = 1,\ldots,d$	**proc RKFn**$(\vec{\phi}, x)$ // $\mathrm{G}_{1,k}$; $k = 1,\ldots,d$
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$	// $\vec{\phi} = (\phi_1,\ldots,\phi_n)$
$\mathsf{T}[0^{i-1} \,\|\, l \,\|\, 0^{n-i}] \leftarrow [a_i^l]$,	// $\phi_i : T_i \to \sum_{j=0}^{d} \alpha_{i,j} \cdot T_i^j$, $\forall i = 1,\ldots,n$
$\quad l = 1,\ldots,k; i = 1,\ldots,n$	If $x \in \mathcal{S}$ then
$\mathsf{T}[0^{i-1} \,\|\, l \,\|\, 0^{n-i}] \xleftarrow{\$} \mathbb{G}$,	$\quad y \xleftarrow{\$} \mathbb{G}$
$\quad l = k+1,\ldots,d; i = 1,\ldots,n$	Else // $x = \omega_i$ for some $i = 1,\ldots,n$
$\mathsf{T}[0^n] \leftarrow [1]$	$\quad y \leftarrow \prod_{l=0}^{d} \mathsf{T}[0^{i-1} \,\|\, l \,\|\, 0^{n-i}]^{\alpha_l^i}$, for $x = \omega_i$
	Return y

Figure 11: Games $\mathrm{G}_{1,k}$ for the proof of Lemma 6.4.

Game $\mathrm{G}_{1,1}$ is identical to Game G_1, since every value in the table is chosen uniformly at random, so we have

$$\Pr\left[\,\textsc{Succ}_1\,\right] = \Pr\left[\,\textsc{Succ}'_{1,1}\,\right].$$

The only difference between games $\mathrm{G}_{1,k}$ and $\mathrm{G}_{1,k+1}$ is the definition of the table values $\mathsf{T}[0^{i-1} \,\|\, k+1 \,\|\, 0^{n-i}]$, for $i = 1,\ldots,n$. Indeed, this value is taken uniformly at random in game $\mathrm{G}_{1,k}$ but set to $\left[a_i^{k+1}\right]$ in game $\mathrm{G}_{1,k+1}$. We design an adversary $\mathscr{D}_k$ against the (n,k)-PDDH problem in $\mathbb{G}$ such that

$$\Pr\left[\,\textsc{Succ}'_{1,k}\,\right] - \Pr\left[\,\textsc{Succ}'_{1,k+1}\,\right] \;\leq\; \mathbf{Adv}_{\mathbb{G}}^{(n,k)\text{-}\mathsf{pddh}}(\mathscr{D}_k)\;.$$

Adversary $\mathscr{D}_k$ does the following. It gets an (n,k)-PDDH tuple $([1], \vec{X}_1, \ldots, \vec{X}_n)$ with for $i = 1,\ldots,n$, $\vec{X}_i = ([a_i], \ldots, [a_i^k], z_i)$ where $a_i \xleftarrow{\$} \mathbb{Z}_p^*$ and either $z_i = \left[a_i^{k+1}\right]$ or $z_i = [c_i]$ for $c_i \xleftarrow{\$} \mathbb{Z}_p^*$. Then $\mathscr{D}_k$ sets $\mathsf{T}[0^n] \leftarrow [1]$ and $\mathsf{T}[0^{i-1} \,\|\, l \,\|\, 0^{n-i}] \xleftarrow{\$} \mathbb{G}$, for $l = k+2,\ldots,d$ and $i = 1,\ldots,n$. It also sets $\mathsf{T}[0^{i-1} \,\|\, l \,\|\, 0^{n-i}] \leftarrow \vec{X}_{i,l} = [a_i^l]$, for $l = 1,\ldots,k$ and $\mathsf{T}[0^{i-1} \,\|\, k+1 \,\|\, 0^{n-i}] \leftarrow \vec{X}_{i,k+1} = z_i$, for $i = 1,\ldots,n$. Hence, if $([1], \vec{X}_1, \ldots, \vec{X}_n)$ is a real (n,k)-PDDH tuple, we have $\mathsf{T}[0^{i-1} \,\|\, k+1 \,\|\, 0^{n-i}] = \left[a_i^{k+1}\right]$, for $i = 1,\ldots,n$, and then we simulate exactly game $\mathrm{G}_{1,k+1}$. If $([1], \vec{X}_1, \ldots, \vec{X}_n)$ is a random (n,k)-PDDH tuple, then $\mathsf{T}[0^{i-1} \,\|\, k+1 \,\|\, 0^{n-i}] = [c_i]$ is a uniformly random value, for all $i = 1,\ldots,n$ and then we simulate exactly game $\mathrm{G}_{1,k}$. This proves the above statement.

To conclude, it is clear that game $\mathrm{G}_{1,d}$ is identical to game G_2.

Equation (3) now follows from equations (6) and (7). □

We now explain how the (N,d)-PDDH assumption and the (N,d)-EDDH assumption are related to the d-DDHI assumption, by introducing two further new assumptions.

***d*-Polynomial DDH.** Let $\mathbb{G}$ be a group of prime order p. For $d \geq 1$, the d-Polynomial Decisional Diffie-Hellman problem in $\mathbb{G}$ consists of deciding, given $([1], [a], \ldots, \left[a^d\right], z)$, with $\mathbb{G} = \langle [1] \rangle$, whether $z = \left[a^{d+1}\right]$ or whether $z = [c]$ for a random $c \in \mathbb{Z}_p^*$. The advantage of an adversary $\mathscr{B}$ against the d-PDDH problem in $\mathbb{G}$, denoted by $\mathbf{Adv}_{\mathbb{G}}^{d\text{-}\mathsf{pddh}}(\mathscr{B})$ is

$$\mathbf{Adv}_{\mathbb{G}}^{d\text{-}\mathsf{pddh}}(\mathscr{B}) = \Pr\left[\,d\text{-PDDH-Real} \Rightarrow 1\,\right] - \Pr\left[\,d\text{-PDDH-Rand} \Rightarrow 1\,\right]$$

where the probabilities are over $a, c \xleftarrow{\$} \mathbb{Z}_p^*$. In particular, this problem is an extension of the Decisional SqDH problem ($d = 1$).

d-Hybrid EDDH. Let $\mathbb{G}$ be a group of prime order p. The d-HEDDH problem in $\mathbb{G}$ consists of deciding, given $(\vec{X}, \vec{Y}) \in \mathbb{G}^{d+1}$, with $X_i = [a^i]_g$, $Y_i = [a^i]_h$ for $i = 0, \ldots, d-1$, whether $X_d = [a^d]_g$, $Y_d = [a^d]_h$, (corresponding to d-HEDDH-Real) or whether $X_d = [c_1], Y_d = [c_2]_h$ are both uniformly random and independent (corresponding to d-HEDDH-Rand). Here $a, c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p^*$, and $g = [1]_g$ and $h = [1]_h$ are random generators of $\mathbb{G}$. The advantage of an adversary $\mathscr{B}$ against the d-HEDDH problem in $\mathbb{G}$, denoted by $\mathbf{Adv}_{\mathbb{G}}^{d\text{-heddh}}(\mathscr{B})$ is

$$\mathbf{Adv}_{\mathbb{G}}^{d\text{-heddh}}(\mathscr{B}) = \Pr[\, d\text{-HEDDH-Real} \Rightarrow 1 \,] - \Pr[\, d\text{-HEDDH-Rand} \Rightarrow 1 \,]$$

where the probabilities are over $a, c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and $g, h \xleftarrow{\$} \mathbb{G} \setminus \{1\}$.

Lemma 6.5. *Let $\mathbb{G}$ be a group of prime order p and $d \geq 2$. Let $\mathscr{A}$ be an adversary against the (N, d)-PDDH problem in $\mathbb{G}$. Then we can construct an adversary $\mathscr{B}$ against the d-PDDH problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}_{\mathbb{G}}^{(N,d)\text{-pddh}}(\mathscr{A}) \;\leq\; N \cdot \mathbf{Adv}_{\mathbb{G}}^{d\text{-pddh}}(\mathscr{B}) \,.$$

The running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time required to compute $(N-1) \cdot d$ exponentiations in $\mathbb{G}$.

Proof of Lemma 6.5. The proof follows a standard hybrid argument. We define games H_j for $j = 0, \ldots, N$ as in Figure 12. Clearly, we have $\mathrm{H}_0 \equiv d$-PDDH-Rand and $\mathrm{H}_N \equiv d$-PDDH-Real. Moreover, it is straightforward to construct an adversary $\mathscr{B}$ such that $\Pr[\, \mathrm{H}_j \Rightarrow 1 \,] - \Pr[\, \mathrm{H}_{j-1} \Rightarrow 1 \,] \leq \mathbf{Adv}_{\mathbb{G}}^{d\text{-pddh}}(\mathscr{B})$, for $j = 1, \ldots, N$. Adversary $\mathscr{B}$ does the following. It gets a d-PDDH tuple $([1], [a], \ldots, [a^d], z)$ and picks a_i, c_i at random in $\mathbb{Z}_p^*$, for $i = 1, \ldots, N$. Then, it set $\vec{X}_i = ([a_i], \ldots, [a_i^{d+1}])$, for $i = 1, \ldots, j-1$ and $\vec{X}_i = ([a_i], \ldots, [a_i^d], [c_i])$, for $i = j+1, \ldots, N$ and finally let $\vec{X}_j = ([a], \ldots, [a^d], z)$, and sends $([1], \vec{X}_1, \ldots, \vec{X}_n)$ to $\mathscr{A}$. The lemma easily follows. □

proc Initialize // H_j, $j = 0, \ldots, N$	**proc Finalize**(b)
$a_i \xleftarrow{\$} \mathbb{Z}_p^*$, for $i = 1, \ldots, N$	Return b
$c_i \xleftarrow{\$} \mathbb{Z}_p^*$, for $i = j+1, \ldots, N$	
$X_{i,k} \leftarrow [a_i{}^k]$, for $k = 1, \ldots, d$, for $i = 1, \ldots, N$	
$X_{i,d+1} \leftarrow [a_i{}^{d+1}]$, for $i = 1, \ldots, j$	
$X_{i,d+1} \leftarrow [c_i]$, for $i = j+1, \ldots, N$	
Return $([1], \vec{X}_1, \ldots, \vec{X}_N)$	

Figure 12: Game for the proof of Lemma 6.5.

Lemma 6.6. *Let $\mathbb{G}$ be a group of prime order p and $d \geq 2$. Let $2 \leq j \leq d$. Let $\mathscr{A}$ be an adversary against the j-PDDH problem in $\mathbb{G}$. Then we can construct an adversary $\mathscr{B}$ against the d-PDDH problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}_{\mathbb{G}}^{j\text{-pddh}}(\mathscr{A}) \;\leq\; \mathbf{Adv}_{\mathbb{G}}^{d\text{-pddh}}(\mathscr{B}) \,.$$

The running time of $\mathscr{B}$ is the same as that of $\mathscr{A}$.

Proof of Lemma 6.6. Adversary $\mathscr{B}$ does the following. It gets a d-PDDH tuple $([1], [a], \ldots, [a^d], z)$. Then, it just sends $([a^{d-j}], [a^{d-j+1}], \ldots, [a^d], z)$ to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. Since $[1] = g$ is a random generator of $\mathbb{G}$ and since a is random in $\mathbb{Z}_p^*$, then $[a^{d-j}]$ is a random generator of $\mathbb{G}$ and $([a^{d-j}], [a^{d-j+1}], \ldots, [a^d], z)$ simulates perfectly a j-PDDH tuple (with generator $[a^{d-j}]$ and exponent a). The lemma easily follows. □

Lemma 6.7. *Let $\mathbb{G}$ be a group of prime order p and $d \geq 2$. Let $(\vec{X}, \vec{Y})$ be a d-*HEDDH *tuple. Then there exists a randomized algorithm R_d that takes $(\vec{X}, \vec{Y}) \in \mathbb{G}^{d+1} \times \mathbb{G}^{d+1}$ as input and outputs $\vec{Y'} \in \mathbb{G}^{d+1}$ with the following properties:*

- *If $(\vec{X}, \vec{Y})$ is a* real *d-*HEDDH *tuple, then so is $(\vec{X}, \vec{Y'})$. Moreover, Y'_0 is uniformly random and independent from $(\vec{X}, \vec{Y})$;*
- *If $(\vec{X}, \vec{Y})$ is a* random *d-*HEDDH *tuple, then so is $(\vec{X}, \vec{Y'})$. Moreover, both Y'_0 and Y'_d are uniformly random and independent from $(\vec{X}, \vec{Y})$ with probability $(1 - \frac{1}{p})$ (over $\vec{X}$ and $\vec{Y}$ only).*

Proof of Lemma 6.7. Let $(\vec{X}, \vec{Y})$ be a d-HEDDH tuple. Let b be the discrete logarithm of $Y_0 = h = [1]_h$ in base $g = [1]_g = X_0$ and let a be the discrete logarithm of X_1 in base g, so that $X_1 = [a]_g$ and $Y_0 = h = [b]_g$, with $a, b \xleftarrow{\$} \mathbb{Z}_p^*$. Then the idea in the algorithm R_d is to randomize the tuple $\vec{Y}$ to produce a new tuple $\vec{Y'}$. We pick $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$ at random. We compute $\vec{Y'}_i \leftarrow Y_i^\alpha \cdot X_i^\beta$ for $i = 0, \ldots, d$. Hence, we have $Y'_0 = h' = [\alpha b + \beta]_g$. For $i = 1, \ldots, d-1$, it is straightforward that $\vec{Y'}_i = [a^i]_{h'}$.

If $(\vec{X}, \vec{Y})$ is a real d-HEDDH tuple, then $Y'_d = \left([ba^d]_g\right)^\alpha \cdot \left([a^d]_g\right)^\beta = [(\alpha b + \beta)a^d]_g = [a^d]_{h'}$. Let $b' = \alpha b + \beta$. Since for any fixed $b', b \in \mathbb{Z}_p^*$ and for any fixed $\alpha \in \mathbb{Z}_p$, there exists a unique $\beta \in \mathbb{Z}_p$ such that $\alpha b + \beta = b'$, it is clear that $Y'_0 = h' = [b']_g$ is uniformly random in $\mathbb{G}$ and independent from $(\vec{X}, \vec{Y})$. Then $(\vec{X}, \vec{Y'})$ is a real d-HEDDH tuple.

Now, if $(\vec{X}, \vec{Y})$ is a random d-HEDDH tuple, then $Y'_d = [c_2]_h{}^\alpha \cdot [c_1]_g{}^\beta = [\alpha b c_2 + \beta c_1]_g$ and we still have $Y'_0 = h' = [\alpha b + \beta]_g$. Here it is not immediately clear that $\vec{Y'}$ is uniformly random and independent from $(\vec{X}, \vec{Y})$. To show this, we fix b, c_1, c_2. Let $b' = \alpha b + \beta$ and $c' = \alpha b c_2 + \beta c_1$. Then $\vec{Y'}$ is uniformly random and independent from $(\vec{X}, \vec{Y})$ if and only if for any fixed $b', c' \in \mathbb{Z}_p^*$, there is a unique $(\alpha, \beta) \in \mathbb{Z}_p$ such that $\alpha b + \beta = b'$ and $\alpha b c_2 + \beta c_1 = c'$. Hence, we need the determinant of the matrix $\begin{pmatrix} b & 1 \\ bc_2 & c_1 \end{pmatrix}$ to be non-zero. This determinant is $D = b(c_1 - c_2)$ so it is non-zero if and only if $b \neq 0$ and $c_1 \neq c_2$. Since $h = [b]_g$ is a generator of $\mathbb{G}$, it is clear that $b \neq 0$. Hence, we have $D \neq 0$ if and only if $c_1 \neq c_2$, which happens with probability $\frac{p-1}{p}$. The claim now follows. □

Lemma 6.8. *Let $\mathbb{G}$ be a group of prime order p and $d \geq 1$. Let $\mathscr{A}$ be an adversary against the (N, d)-*EDDH *problem in $\mathbb{G}$. Then we can construct adversaries $\mathscr{B}_j$ against the j-*HEDDH *problem in $\mathbb{G}$, for $j = 1, \ldots, d$ such that*

$$\mathbf{Adv}_{\mathbb{G}}^{(N,d)\text{-eddh}}(\mathscr{A}) \;\leq\; \sum_{j=1}^{d} \frac{p}{p-1} \mathbf{Adv}_{\mathbb{G}}^{j\text{-heddh}}(\mathscr{B}_j) \;\leq\; \frac{p}{p-1} \cdot d \cdot \mathbf{Adv}_{\mathbb{G}}^{d\text{-heddh}}(\mathscr{B}_d)\,.$$

The running time of $\mathscr{B}_j$ is that of $\mathscr{A}$ plus the time required to compute $(N-1)\cdot 2\cdot(j+1)$ exponentiations in $\mathbb{G}$.

Proof of Lemma 6.8. The proof follows a standard hybrid argument. We define games H_j for $j = 0, \ldots, d$ as in Figure 13. Clearly, we have $\mathrm{H}_0 \equiv (N, d)$-EDDH-Rand and $\mathrm{H}_d \equiv (N, d)$-EDDH-Real. Moreover, it is straightforward to construct an adversary $\mathscr{B}_j$ such that $\Pr[\mathrm{H}_j \Rightarrow 1] - \Pr[\mathrm{H}_{j-1} \Rightarrow 1] \leq$

$\mathbf{Adv}_{\mathbb{G}}^{j\text{-heddh}}(\mathscr{B}_j)$, for $j = 1, \ldots, d$. Adversary $\mathscr{B}_j$ just executes $N-1$ times algorithm R_j to compute $Z_{k,l}$ for $k = 2, \ldots, N$ and $l = 0, \ldots, j$, using the j-HEDDH tuple it gets as input, so its running time is that of $\mathscr{A}$ plus the time to compute $(N-1) \cdot 2 \cdot (j+1)$ exponentiations in $\mathbb{G}$. Moreover, the simulation is perfect with probability $1 - \frac{1}{p}$.

Finally, it is clear that for $j = 1, \ldots, d$, $\mathbf{Adv}_{\mathbb{G}}^{j\text{-heddh}}(\mathscr{B}_j) \leq \mathbf{Adv}_{\mathbb{G}}^{d\text{-heddh}}(\mathscr{B}_d)$, since from a d-HEDDH tuple $(\vec{X}, \vec{Y})$, with $\vec{X}_i = \left[a^i\right]_g$, $Y_i = \left[a^i\right]_h$ for $i = 0, \ldots, d-1$, and with $X_d = \left[a^d\right]_g$ and $Y_d = \left[a^d\right]_h$ or $X_d = [c_1]_g$ and $Y_d = [c_2]_h$ where $a, c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and g and h are random generators of $\mathbb{G}$, we can extract a j-HEDDH tuple using the last $j+1$ components of $\vec{X}$ and $\vec{Y}$ (generators are now $\left[a^{d-j}\right]_g$ and $\left[a^{d-j}\right]_h$, which are random generators of $\mathbb{G}$ since g, h are random generators of $\mathbb{G}$ and since a is random). The lemma easily follows. □

proc Initialize // H_j, $j = 0, \ldots, d$	**proc Finalize**(b)
$a_0 \leftarrow 1$; $a_k, b \xleftarrow{\$} \mathbb{Z}_p^*$, for $k = 1, \ldots, N$	Return b
$c_{k,l} \xleftarrow{\$} \mathbb{Z}_p^*$, for $l = j+1, \ldots, d$ and $k = 0, \ldots, N$	
$Z_{k,l} \leftarrow \left[a_k b^l\right]$, for $l = 0, \ldots, j$ and $k = 0, \ldots, N$	
$Z_{k,l} \leftarrow [c_{k,l}]$, for $l = j+1, \ldots, d$ and $k = 0, \ldots, N$	
Return $\vec{Z}$	

Figure 13: Game for the proof of Lemma 6.8.

Lemma 6.9. *Let $\mathbb{G}$ be a group of prime order p and $d \geq 1$. Let $\mathscr{A}$ be an adversary against the d-HEDDH problem in $\mathbb{G}$. Then, we can construct an adversary $\mathscr{B}$ against the d-PDDH problem in $\mathbb{G}$, such that*

$$\mathbf{Adv}_{\mathbb{G}}^{d\text{-heddh}}(\mathscr{A}) \leq \frac{p}{p-1} \mathbf{Adv}_{\mathbb{G}}^{d\text{-pddh}}(\mathscr{B}) .$$

The running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time required to compute $2 \cdot (d+1)$ exponentiations in $\mathbb{G}$.

Proof of Lemma 6.9. Let $\mathbb{G}$ be a group of prime order p and $d \geq 1$. Adversary $\mathscr{B}$ gets a d-PDDH tuple $\vec{Z} = ([1], [a], \ldots, \left[a^d\right], z) = (Z_0, \ldots, Z_{d+1}) \in \mathbb{G}^{d+2}$ and set $X_i \leftarrow Z_{i+1}$, for $i = 0, \ldots, d$, so $\vec{X} = ([a], \ldots, \left[a^d\right], z)$. Next, $\mathscr{B}$ does the following. It first chooses $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$ at random and computes the tuple $\vec{Y}$ by letting $Y_l \leftarrow Z_{l+1}^{\alpha} \cdot Z_l^{\beta}$ for $l = 0, \ldots, d$. Hence, for $l = 0, \ldots, d-1$, it is clear that $Y_l = \left[(\alpha a + \beta) \cdot a^l\right]$.

If $Z_{d+1} = \left[a^{d+1}\right]$, we have $Y_d = \left[(\alpha a + \beta) \cdot a^d\right]$ and $X_d = \left[a \cdot a^d\right]$. Hence, since g and $a \in \mathbb{Z}_p^*$ are random, $X_0 = [a]$ is a random generator of $\mathbb{G}$ and $Y_0 = [\alpha a + \beta]$ is uniformly random and independent from X_0 (these claims follow using similar arguments to those in the proof of Lemma 6.7) and $(\vec{X}, \vec{Y})$ is a real d-HEDDH tuple.

If $Z_{d+1} = [c]$ with $c \xleftarrow{\$} \mathbb{Z}_p^*$, then $Y_d = \left[\alpha c + \beta a^d\right]$. We fix $a, c \in \mathbb{Z}_p^*$ and we let $b' = \alpha a + \beta$ and $c' = \alpha c + \beta a^d$. Then, Y_d is uniformly random and independent from X_d if and only if for any fixed $b', c' \in \mathbb{Z}_p^*$, there is a unique $(\alpha, \beta) \in \mathbb{Z}_p$ such that $b' = \alpha a + \beta$ and $c' = \alpha c + \beta a^d$. Hence, we need the determinant of the matrix $\begin{pmatrix} a & 1 \\ c & a^d \end{pmatrix}$ to be non-zero. This determinant is $D = a^{d+1} - c$ so it is non-zero if and only if $c \neq a^{d+1}$. Since c is by definition uniformly random in $\mathbb{Z}_p$, we have $D \neq 0$ with probability $\frac{p-1}{p}$. The claim easily follows. □

Lemma 6.10. *Let $\mathbb{G}$ be a group of prime order p and $d \geq 1$. Let $\mathscr{A}$ be an adversary against the d-PDDH problem in $\mathbb{G}$. Then we can construct an adversary $\mathscr{B}$ against the d-DDHI problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}^{d\text{-}\mathsf{pddh}}_{\mathbb{G}}(\mathscr{A}) \;\leq\; \mathbf{Adv}^{d\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{B})\,.$$

Moreover, the running time of $\mathscr{B}$ is the same as $\mathscr{A}$.

Proof of Lemma 6.10. Let $\mathbb{G}$ be a group of prime order p and $d \geq 1$. $\mathscr{B}$ gets a d-DDHI tuple $([1], [a], \ldots, [a^d], z) \in \mathbb{G}^{d+2}$. Then, $\mathscr{B}$ runs $\mathscr{A}$, giving it the tuple $([a^d], [a^{d-1}], \ldots, [a], [1], z)$. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. Since $g = [1]$ and a are random, $h = [a^d]$ is a random generator of $\mathbb{G}$ and $\frac{1}{a}$ is random in $\mathbb{Z}_p^*$, so we have $[a^{d-j}] = [(\frac{1}{a})^j]_h$, for $j = 0, \ldots, d$ and if $([1], [a], \ldots, [a^d], z)$ is a real d-DDHI tuple then z is equal to $[\frac{1}{a}] = [(\frac{1}{a})^{d+1}]_h$, otherwise z is random in $\mathbb{G}$. Hence, the simulation is perfect and the claim follows. □

Lemma 6.3 now follows from combining Lemmas 6.4–6.10. □

Finally, by combining the results in Lemmas 6.1–6.3 with Theorem 5.1, we can prove the following theorem.

Theorem 6.11. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let NR^* be defined via $\mathsf{NR}^*(\vec{a}, x) = \left[\prod_{i=1}^n a_i^{x[i]}\right]$, where $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n \setminus \{0^n\}$. Let $\overline{\mathcal{D}} = \{0,1\}^n \times \mathbb{G}^n$ and let $h\colon\ \overline{\mathcal{D}} \to \{0,1\}^{n-2}$ be a hash function. Let $\omega_i = 0^{i-1} \,\|\, 1 \,\|\, 0^{n-i}$, for $i = 1, \ldots, n$. Define $F\colon\ \mathbb{Z}_p^n \times \{0,1\}^n \to \mathbb{G}$ by*

$$F(\vec{a}, x) = \mathsf{NR}^*(\vec{a}, 11 \,\|\, h(x, \mathsf{NR}^*(\vec{a}, \vec{\omega})))$$

for all $\vec{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n$. Let $\mathscr{A}$ be a Φ_d-restricted adversary against the prf-rka security of F that makes $Q_{\mathscr{A}} \leq |\{0,1\}^{n-2}|$ oracle queries. Then, assuming $nd \leq \sqrt{p}$, we can construct an adversary $\mathscr{B}$ against the d-DDHI problem in $\mathbb{G}$, an adversary $\mathscr{C}$ against the cr security of h, and an adversary $\mathscr{D}$ against the d-SDL problem in $\mathbb{G}$ such that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_d,F}(\mathscr{A}) \;\leq\; &\left(n \cdot d \cdot (1-1/p)^2 + n \cdot (d-1)\right) \cdot \mathbf{Adv}^{d\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{B}) \\ &+ \mathbf{Adv}^{\mathsf{cr}}_{h}(\mathscr{C}) + n \cdot \mathbf{Adv}^{d\text{-}\mathsf{sdl}}_{\mathbb{G}}(\mathscr{D}) + \left(d \cdot Q_{\mathscr{A}}^2 + 4n \cdot Q_{\mathscr{A}}\right)/(2p).\end{aligned}$$

The running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus $O(d \cdot (n + Q_{\mathscr{A}}))$ exponentiations in $\mathbb{G}$ and $O(Q_{\mathscr{A}}^3 \cdot (nd + Q_{\mathscr{A}}))$ operations in $\mathbb{Z}_p$. $\mathscr{C}$ has the same running time as $\mathscr{A}$. The running time of $\mathscr{D}$ is that of $\mathscr{A}$ plus the time required to factorize a polynomial of degree at most d in $\mathbb{F}_p$, which is sub-quadratic in d, logarithmic in p.

Acknowledgements

We thank Susan Thomson for bringing the issues in the original Bellare-Cash framework to our attention, and for useful comments on the paper. Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue were supported by the French ANR-10-SEGI-015 PRINCE Project, the *Direction Générale de l'Armement* (DGA), the CFM Foundation, the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet, and the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013 Grant Agreement 339563 – CryptoCloud). Kenneth G. Paterson was supported by an EPSRC Leadership Fellowship, EP/H005455/1.

A From Malleability to Unique-Input-RKA-Security

In this section, we explore the relationship between key-malleable PRFs and unique-input RKA secure PRFs. Specifically, we show that the $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of a Φ-key-malleable PRF M is implied by its regular prf security if the key-transformer KT associated with M satisfies a new notion of uniformity that we call $\mathcal{S}$-uniformity and which is defined below.

$\mathcal{S}$-Uniform Key-Transformer. Let $M\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions and Φ be a class of RKD-functions, such that there is a key-transformer KT for (M, Φ). We generalize the uniformity property of a key-transformer, defined in [BC10a, Section 3.1], by allowing the uniformity condition to be restricted to a subset $\mathcal{S}$ of $\mathcal{D}$. Indeed, let $\mathcal{S}$ be a subset of $\mathcal{D}$; then we say that KT is an *$\mathcal{S}$-Uniform Key-Transformer* if the games $\mathcal{S}$-KTReal and $\mathcal{S}$-KTRand defined in Figure 14 are *perfectly indistinguishable* for any Φ-restricted adversary $\mathscr{A}$, where $\mathscr{A}$ belongs to the class of adversaries such that all queries (ϕ, x) with $x \in \mathcal{S}$ made by $\mathscr{A}$ to its oracle are for distinct values of x. That is,

$$\Pr\left[\mathcal{S}\text{-KTReal}_{\mathsf{T}}^{\mathscr{A}} \Rightarrow 1\right] = \Pr\left[\mathcal{S}\text{-KTRand}_{\mathsf{T}}^{\mathscr{A}} \Rightarrow 1\right].$$

Note that the standard uniformity condition given in Section 2 corresponds to the KT being a $\mathcal{D}$-uniform key-transformer.

proc Initialize // $\mathcal{S}$-KTReal	**proc Initialize** // $\mathcal{S}$-KTRand
$K \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \mathcal{K}$
$f \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$	$f \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$
proc RKFn(ϕ, x) // $\mathcal{S}$-KTReal	**proc RKFn**(ϕ, x) // $\mathcal{S}$-KTRand
$y \leftarrow \mathsf{KT}^f(\phi, x)$	If $x \in \mathcal{S}$ then $y \xleftarrow{\$} \mathcal{R}$
Return y	Else $y \leftarrow \mathsf{KT}^f(\phi, x)$
proc Finalize(b') // $\mathcal{S}$-KTReal	Return y
Return b'	**proc Finalize**(b') // $\mathcal{S}$-KTRand
	Return b'

Figure 14: Games used in the definition of an $\mathcal{S}$-uniform key-transformer KT.

Theorem A.1. *Let $M\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions and Φ be a class of* RKD*-functions, such that there exists an $\mathcal{S}$-uniform key-transformer KT for (M, Φ). Let $\mathscr{A}$ be an adversary against the $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of M that makes $Q_{\mathscr{A}}$ oracle queries. Then we can design an adversary $\mathscr{B}$ against the standard prf security of M such that*

$$\mathbf{Adv}_M^{(\mathcal{S},\Phi)\text{-ui-prf-rka}}(\mathscr{A}) \leq 2 \cdot \mathbf{Adv}_M^{\mathsf{prf}}(\mathscr{B}). \tag{8}$$

Moreover, the running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time required to execute $Q_{\mathscr{A}}$ times the key-transformer.

Proof of Theorem A.1. The proof is based on the sequence of games in Figure 15. Let $\textsc{Succ}_i$ denote the event that game G_i output takes the value 1.

In game G_1, we use the key-transformer KT to compute $M(\phi(K), \cdot)$ via oracle calls to $M(K, \cdot)$. The correctness property of the key transformer implies

$$\Pr[\textsc{Succ}_0] = \Pr[\textsc{Succ}_1].$$

In game G_2, we replace the oracle $M(K,\cdot)$ given to the key transformer KT by a random function f. We design an adversary $\mathscr{B}$ attacking the standard prf security of M such that

$$\Pr[\,\text{SUCC}_1\,] \le \Pr[\,\text{SUCC}_2\,] + \mathbf{Adv}^{\mathsf{prf}}_M(\mathscr{B})\ .$$

Adversary $\mathscr{B}$ runs $\mathscr{A}$. When the latter makes an RKFn-query (ϕ, x), adversary $\mathscr{B}$ responds via

$y \leftarrow \mathsf{KT}^{\mathbf{Fn}}(\phi, x)$
Return y

where **Fn** is $\mathscr{B}$'s own oracle. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. Then

$$\Pr\left[\,\text{PRFReal}^{\mathscr{B}}_M \Rightarrow 1\,\right] = \Pr[\,\text{SUCC}_1\,] \quad \text{and} \quad \Pr\left[\,\text{PRFRand}^{\mathscr{B}}_M \Rightarrow 1\,\right] = \Pr[\,\text{SUCC}_2\,]\ .$$

Games G_2 and G_3 differ only in the way that queries (ϕ, x) with $x \in \mathcal{S}$ are handled. Indeed, for such queries, in G_3, the output is just taken uniformly at random instead of computed using the key-transformer. Since the adversary $\mathscr{A}$ belongs to the class of Φ-restricted adversaries such that all queries (ϕ, x) with $x \in \mathcal{S}$ made by $\mathscr{A}$ to its oracle are for distinct values of x, games G_2 and G_3 match exactly games $\mathcal{S}$-KTReal and $\mathcal{S}$-KTRand, respectively. Since we assume that KT is an $\mathcal{S}$-uniform key-transformer KT, these two games are perfectly indistinguishable. So

$$\Pr[\,\text{SUCC}_2\,] = \Pr[\,\text{SUCC}_3\,]\ .$$

In game G_4, we replace the random function f given to the key-transformer KT by the oracle $M(K,\cdot)$. We design an adversary $\mathscr{B}$ attacking the prf security of M such that

$$\Pr[\,\text{SUCC}_3\,] \le \Pr[\,\text{SUCC}_4\,] + \mathbf{Adv}^{\mathsf{prf}}_M(\mathscr{B}).$$

Adversary $\mathscr{B}$ runs $\mathscr{A}$. When the latter makes an RKFn-query (ϕ, x), adversary $\mathscr{B}$ responds via

If $x \in \mathcal{S}$ then $y \xleftarrow{\$} \mathcal{R}$
Else $y \leftarrow \mathsf{KT}^{\mathbf{Fn}}(\phi, x)$
Return y

where **Fn** is $\mathscr{B}$'s own oracle. When $\mathscr{A}$ halts with output b, $\mathscr{B}$ halts with output $1-b$. Then

$$\Pr\left[\,\text{PRFReal}^{\mathscr{B}}_M \Rightarrow 1\,\right] = \Pr[\,\text{SUCC}_3\,] \quad \text{and} \quad \Pr\left[\,\text{PRFRand}^{B}_M \Rightarrow 1\,\right] = \Pr[\,\text{SUCC}_4\,]\ .$$

Finally, in game G_5, instead of using the key-transformer KT to compute $M(\phi(K),\cdot)$ via oracle calls to $M(K,\cdot)$, we use M directly. The correctness property of the key transformer implies that

$$\Pr[\,\text{SUCC}_4\,] = \Pr[\,\text{SUCC}_5\,]\ .$$

Equation (8) on 35 now follows by combining the bounds arising in the different game hops. □

Application to NR^*. In what follows, we apply Theorem A.1 and Theorem 5.1 to NR^* to prove that the construction given in Section 4 is a Φ_{aff}-RKA-secure PRF. Note that this gives an alternative proof of Theorem 4.5.

Let $\mathcal{S} = \{0,1\}^n \setminus (\{\omega_1, \ldots, \omega_n\} \cup \{0^n\})$. The only point that remains to prove is that there exists an $\mathcal{S}$-uniform key-transformer $\mathsf{KT}_{\Phi_{\mathsf{aff}}}$ for NR^* and the class Φ_{aff} of RKD functions. This result is actually implied by Lemma 4.3. Indeed, the same key-transformer is an $\mathcal{S}$-uniform key-transformer for $(\mathsf{NR}^*, \Phi_{\mathsf{aff}})$. This statement is implied by the fact that games G_0 and G_{n-1}, defined in the proof

proc Initialize // G_0 $K \xleftarrow{\$} \mathcal{K}$ **proc RKFn**(ϕ, x) // G_0 $y \leftarrow M(\phi(K), x)$ Return y **proc Finalize**(b') // All Games Return b'	**proc Initialize** // G_1 $K \xleftarrow{\$} \mathcal{K}$ **proc RKFn**(ϕ, x) // G_1 $y \leftarrow \mathsf{KT}^M(\phi, x)$ Return y	**proc Initialize** // G_2 $K \xleftarrow{\$} \mathcal{K}$ $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ **proc RKFn**(ϕ, x) // G_2 $y \leftarrow \mathsf{KT}^f(\phi, x)$ Return y
proc Initialize // G_3 $K \xleftarrow{\$} \mathcal{K}$ $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ **proc RKFn**(ϕ, x) // G_3 If $x \in \mathcal{S}$ then $y \xleftarrow{\$} \mathcal{R}$ Else $y \leftarrow \mathsf{KT}^f(\phi, x)$ Return y	**proc Initialize** // G_4 $K \xleftarrow{\$} \mathcal{K}$ **proc RKFn**(ϕ, x) // G_4 If $x \in \mathcal{S}$ then $y \xleftarrow{\$} \mathcal{R}$ Else $y \leftarrow \mathsf{KT}^M(\phi, x)$ Return y	**proc Initialize** // G_4 $K \xleftarrow{\$} \mathcal{K}$ **proc RKFn**(ϕ, x) // G_4 If $x \in \mathcal{S}$ then $y \xleftarrow{\$} \mathcal{R}$ Else $y \leftarrow M(\phi(K), x)$ Return y

Figure 15: Games for the proof of Theorem A.1.

of Lemma 4.3, are indistinguishable, even when the adversary is not a unique-input adversary with respect to the set $\overline{\mathcal{S}}$ (where $\overline{\mathcal{S}}$ denotes the complement of $\mathcal{S}$).

To see why, note that the argument used to prove that Games G_j and G_{j+1} are indistinguishable in that proof remains valid because all points in $\overline{\mathcal{S}}$ have a strictly smaller Hamming weight than those in $\mathcal{S}$. Hence, there exists an $\mathcal{S}$-uniform key-transformer $\mathsf{KT}_{\Phi_{\mathsf{aff}}}$ for $(\mathsf{NR}^*, \Phi_{\mathsf{aff}})$, and the $(\mathcal{S}, \Phi_{\mathsf{aff}})$-unique-input-prf-rka security of NR^* follows. Finally, by applying Theorem 5.1, we can prove a similar statement to the one in Theorem 4.5.

Remark A.2. It is worth noting that it is not clear that the usual uniformity condition of a key-transformer directly implies that the same key-transformer is $\mathcal{S}$-uniform for $\mathcal{S} \subset \mathcal{D}$ without making additional assumptions about the key-transformer.

References

[BC10a] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, August 2010. (Cited on pages 1, 2, 3, 4, 6, 7, 8, 9, 12, 13, 18, and 35.)

[BC10b] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. Cryptology ePrint Archive, Report 2010/397, 2010. http://eprint.iacr.org/, last updated 27/10/2013. (Cited on pages 2 and 7.)

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 486–503. Springer, December 2011. (Cited on pages 3 and 4.)

[BDK05] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 507–525. Springer, May 2005. (Cited on page 1.)

[BDK08] Eli Biham, Orr Dunkelman, and Nathan Keller. A unified approach to related-key attacks. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 73–96. Springer, February 2008. (Cited on page 1.)

[BDK+10] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 299–319. Springer, May 2010. (Cited on page 1.)

[Bih94] Eli Biham. New types of cryptoanalytic attacks using related keys (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 398–409. Springer, May 1994. (Cited on page 1.)

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, May 2003. (Cited on pages 1 and 6.)

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer, December 2009. (Cited on page 1.)

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, August 2009. (Cited on page 1.)

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, August 2013. (Cited on page 3.)

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, August 2014. (Cited on page 3.)

[BPT12] Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: IBE, encryption and signatures. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 331–348. Springer, December 2012. (Cited on page 3.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. (Cited on page 6.)

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, August 2013. (Cited on pages 1 and 5.)

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984. (Cited on page 6.)

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 182–200. Springer, March 2011. (Cited on pages 3, 4, and 7.)

[KHP07] Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-key rectangle attacks on reduced AES-192 and AES-256. In Alex Biryukov, editor, *FSE 2007*, volume 4593 of *LNCS*, pages 225–241. Springer, March 2007. (Cited on page 1.)

[Knu93] Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 196–208. Springer, December 1993. (Cited on page 1.)

[KU11] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802, 2011. (Cited on page 22.)

[LMR14] Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Improved constructions of PRFs secure against related-key attacks. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14*, volume 8479 of *LNCS*, pages 44–61. Springer, June 2014. (Cited on page 3.)

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997. (Cited on page 1.)

[Wee12] Hoeteck Wee. Public key encryption against related key attacks. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 262–279. Springer, May 2012. (Cited on page 3.)